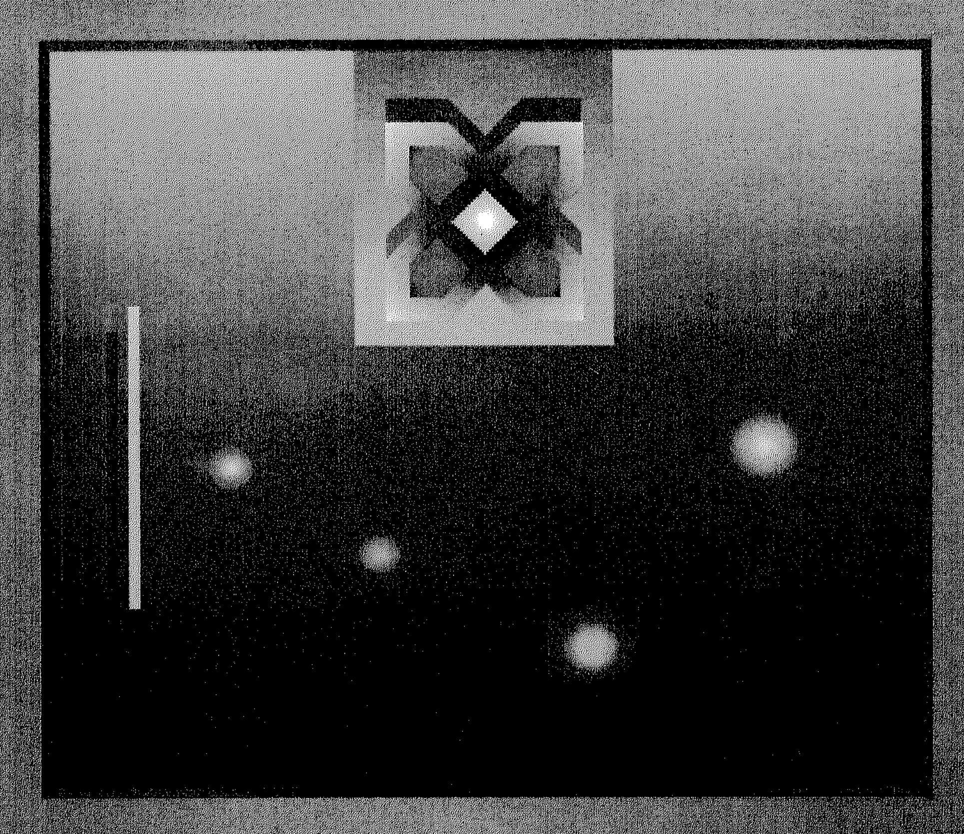
الطبعة الثانية

الحاسب وتطبيقات نظم إدارة قواعد البيانات



م. مصطفى رضا عبد الوهاب — أ.د. محمد على الشرقاوى

...دين محمد فهمى — مصطفى محمد اسماعيل

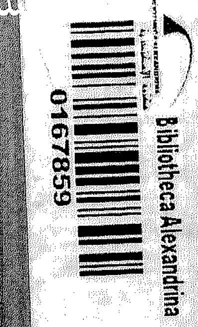
تحقيق وتقديم

أ.د. محمد فهمى طلبه

٧

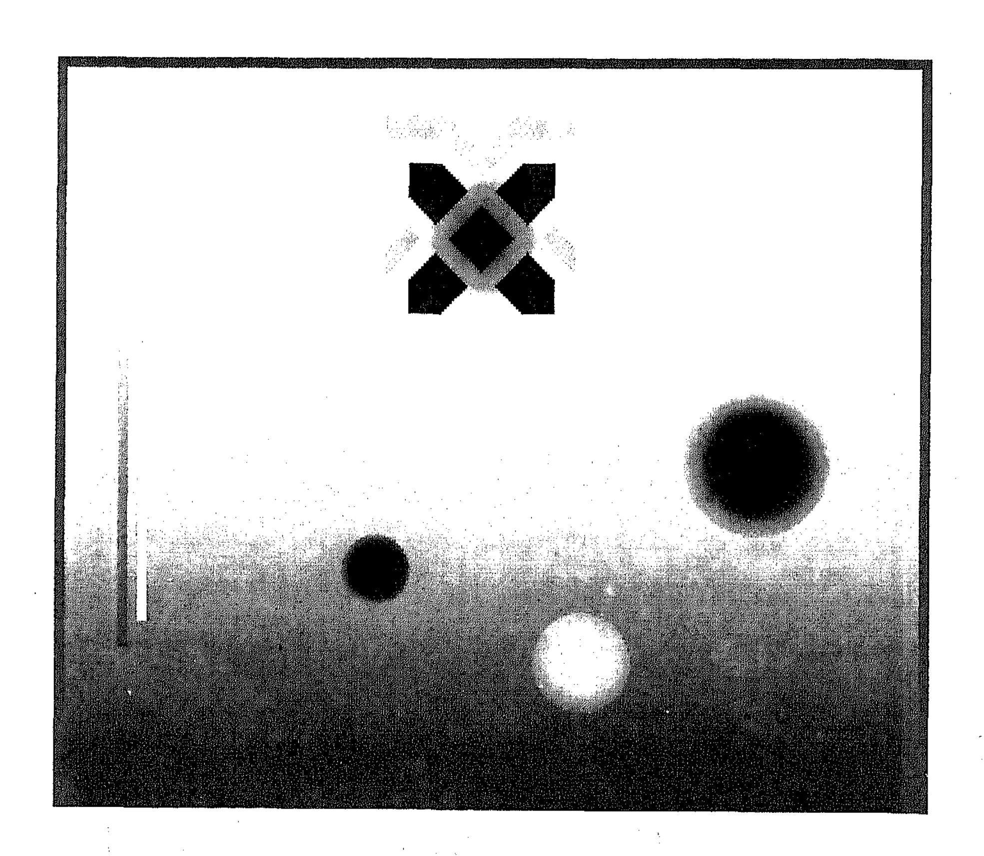


مجموعة كتب دلتا



الحاسب وتطبيقات نظم إدارة قواعد البيانات

الحاسب وتطبيقات نظم إدارة قواعد البيانات



م.مصطفى رضا عبد الوهاب   ا.د.محمد على الشرقاوى

د. علاء الدين محمد فهمى   مصطفى محمد اسماعيل

تحقيق وتقديم

ا.د. محمد فهمى طلبه

٧

مجموعة كتب دلتا

تقـــديــم

(الطبعـة الثـانيـة)

إن السؤال الدائر بين المتخصصين فى مجال الحاسبات عن كون تخطيط البرامج على وجه الخصوص وبعض جوانب علوم الحاسب عامة تقع فى إطار العلم (Science) أو فى إطار الفن (Art). وبدون الخوض فى تفاصيل تعريف كل منهما من ناحية الأسس القائم عليها وعلاقة ذلك بالإنسان فإنه يوجد شبه إتفاق بين كل الخبراء فى أن تخطيط البرامج وإن كان يرتكز على العديد من الأسس العلمية التى يجب الإلمام بها إلا أنه يحتاج إلى العديد من الجوانب المرتبطة بالفن والتى تتأثر بخبرة مخطط البرامج وقدرته الإبداعية بالإضافة إلى شخصيته.

ومما لاشك فيه أن البرنامج الواحد - الذى يؤدى مهمة معينة للوصول إلى هدف أو أهداف محددة - إذا ما تم تنفيذه بواسطة العديد من مخططى البرامج فإنه يصعب أن تتفق خطوات التنفيذ لأى منهم مع الآخر رغم أن النظام النهائى يؤدى نفس المهمة. ولا يعتبر الاختلاف فى النظم المنتجة ناشئا فقط من التباين فى الخلفية العلمية لمخططى البرامج بقدر ما يعبر عن التباين فى خبرتهم وشخصيتهم بالإضافة إلى قدرتهم الإبداعية.

ولاخلاف فى أن الخبرة والقدرة الإبداعية يمكن أن تصقل بواسطة الممارسة بعد التقليد. ومن هذا البعد كانت فكرة هذا الكتاب الذى يعتبر أحد أهدافه تمكين مخطط البرامج - الملم بكل الجوانب العلمية لهندسة تخطيط البرامج والمتمكن من أدوات البرمجة المستخدمة - من متابعة بعض النماذج للحلول المقترحة لمشاكل معينة بحيث تمكنه بصورة أو بأخرى من تقليدها فى أحد الإطارات الأخرى المشابهة. ومع التباين بين النظم التى تم مناقشتها يستطيع القارىء إكتساب بعض الخبرات فى العديد من المجالات المطروحة فى هذا الكتاب.

ويمكن للقارىء أن ينتقل من مرحلة الإستيعاب إلى مرحلة التقليد ثم إلى مرحلة التعديل يليها مرحلة التطوير ثم ينطلق إلى مرحلة الإبداع. وقد اختار هذا الكتاب أحد أدوات البرمجة المتميزة وهى برامج عائلة (DBASE). وتنتمى إلى هذه العائلة

نظم برامج (DBASE III+, DBASE IV, CLIPPER, FOXBASE, FOXPRO) ولابد للقارىء عند الإطلاع على هذا الكتاب أن يكون ملما بأوامر وأساليب البرمجة المستخدمة فى لغة هذه العائلة. وجدير بالذكر أن كتاب نظم إدارة قواعد البيانات " الجزء الأول " و " الجزء الثانى" من مجموعة كتب دلتـا تمكن القارىء المبتدىء من التعرف على أوامر وأساليب البرمجة باستخدام أى من نظم برامج عائلة (DBASE).

وفى مرحلة الإستيعاب يمكن للقارىء أن يركز فى مضمون النظم الموضحة فى أجزاء وفصول الكتاب المختلفة ، وأن يحدد وظيفة كل جزء من هذه الأجزاء ومن ثم التعرف على بعض الأفكار المطروحة والإطار العام لتصميم النظام وهيكله البنائى.

أما فى مرحلة التقليد فإنه يمكن للقارىء أن يستخدم نفس المنهج المحدد فى هذا الكتاب لبناء نظم مشابهة. وعندما يتمكن القارىء من كل مفاتيح عملية تخطيط البرامج يستطيع اجراء التعديلات المختلفة على الأمثلة المحددة فى هذا الكتاب بما يتلائم مع احتياجات نظامه وبما يمكنه من الوصول الى الهدف المطلوب. عند هذه المرحلة يكون القارىء قد بنى خبرة جيدة ومتميزة فى إنتاج البرامج باستخدام نظم عائلة (DBASE) يتمكن بعدها من تطوير العديد من البرامج المشابهة والغير مشابهة للنظم المذكورة فى هذا الكتاب ثم يلى ذلك مرحلة الإبداع حيث يستطيع القارىء إيجاد أساليب مختلفة تعبر عن شخصيته وتميز أسلوبه وقدرته على الإبتكار.

ويتكون الكتاب من ثمانى وعشرين فصلا مقسمة إلى ستة أجزاء. الجزء الأول منها مراجعة شاملة لنظم برامج عائلة (DBASE). أما الجزء الثانى فيوضح نظام معلومات لشئون الطلبة الذى يصلح للاستخدام فى العديد من المدارس وبعض كليات الجامعة بطريقة مبسطة وسهلة. ويعتبر الجزء الثالث أحد البرامج المحاسبية الشائعة الاستخدام وهو يوضح نظام المخازن. أما الجزء الرابع فهو متعلق بنظام حسابات العملاء وهو أيضا من البرامج المحاسبية المستخدمة فى العديد من المؤسسات التجارية والصناعية. أما الجزء الخامس فإنه يضيف بعض الأساليب المتقدمة لتخطيط البرامج والتى يمكن أن يستفيد منها مخطط البرامج

فى رفع كفاءته وقدرته على استخدام نظم برامج (DBASE). والجزء السادس يشرح بعض التطبيقات الإضافية والتى تتضمن تطبيقا منزليا يمكن استخدامه فى تخزين بيانات المعارف والأقارب وإسترجاع وتعديل أو مسح أى بيان. كما يتضمن هذا الجزء أيضا شرح استخدام مولد التطبيقات (Application Generator) الخاص ببرنامج (DBase III+).

وجدير بالذكر أنه سبق طباعة النسخة الأولى من هذا الكتاب وهذه هى الطبعة الثانية التى روعى فيها إضافة مزيد من التطبيقات وكذلك إضافة معلومات عن مولد التطبيقات باعتباره من أحدث وسائل تخطيط البرامج.

ويعتبر هذا الكتاب أحد المحاولات الجيدة التى يمكن أن توضح للدارس بعض النظم التطبيقية الشائعة الاستخدام. ومن هذا البعد - بالإضافة ما سبق ذكره - فإن هذا الكتاب يعتبر إضافة حقيقية للمكتبة العربية فى مجال تكنولوچيا وعلوم الحاسب.

ا . د . محمد فهمى طلبه

محتويات الكتاب

مسلسل	الموضوع	رقم الصفحة

مسلسل	الموضوع	رقم الصفحة

مسلسل	الموضوع	رقم الصفحة

مسلسل	الموضوع	رقم الصفحة

مسلسل　　　　الموضوع　　　　رقم الصفحة

مسلسل | الموضوع | رقم الصفحة

مسلسل	الموضوع	رقم الصفحة

مسلسل	الموضوع	رقم الصفحة

الفصل الأول

مقـــدمـــة

هذا الكتاب يمثل التطبيق العملى للكتابين السابقين (الكتاب رقم (٥) والكتاب رقم (٦) من مجموعة كتب دلتا) على نظم المعلومات الشائعة الإستخدام مثل نظم معلومات شئون الطلبة (Cadets) ونظم المخازن (Inventory) ونظم حسابات العملاء (Accounts Receivable)، وهو يركز على أحدث الوسائل (Techniques) التى تستخدم فى كتابة البرامج القوية ذات الكفاءة العالية وسرعة التشغيل الكبيرة.

والكتاب لايكتفى بعرض البرامج ولكنه يشرح كل برنامج خطوة خطوة حتى يستوعب القارىء البرامج. كما أنه يراعى استخدام معظم أوامر عائلة (DBase) والدوال الخاصة بها واستخدام كل البدائل الممكنة حتى يصبح القارىء ملما بجميع إمكانيات البرنامج، كما تتوفر لديه القدرة على كتابة برامج كاملة قابلة للتنفيذ. كما توفر مؤسسة دلتا أقراصا تحتوى على هذه البرامج بحيث يستطيع القارئ - الذى لايجد وقتا كافيا لكتابتها - استخدامها مباشرة من الأقراص مع إمكانية تتبع سطور البرنامج ومراجعتها قبل تنفيذها.

وهذا الكتاب كما سبق الإيضاح هو استكمال للكتابين السابقين ، لذلك فمن المفيد قراءة هذين الكتابين حتى يستطيع القارىء استيعاب البرامج الموجودة. ويستطيع القارىء كتابة هذه البرامج على الحاسب وتنفيذها حيث أن هذا يكسبه الخبرة المطلوبة لكتابة أى برامج أخرى.

والكتاب ينقسم إلى ستة أجزاء ، الجزء الأول عبارة عن مراجعة شاملة لبرامج عائلة (DBase) والأوامر المستخدمة فيها مع شرح لطرق كتابة البرامج وبعض الوسائل التى تزيد من كفاءتها وسرعة تنفيذها.

والجزء الثانى يشرح برنامج نظام معلومات شئون الطلبة (Cadets) الذى يمثل نوعا من البرامج التى تتعامل مع قاعدة بيانات واحدة (Single Database). وهو برنامج يصلح للمبتدئين حيث أنه يركز على أساسيات كتابة البرامج التى تعتمد على القوائم فى تشغيلها بواسطة المستخدم (User Friendly). كما يركز أيضا على تصميم شاشات الإدخال وتصميم التقارير.

والجزء الثالث يشرح برنامج المخازن (Inventory) الذى يوضح اساسيات التعامل مع عدة ملفات قواعد بيانات وربط هذه الملفات وفتحها من مناطق عمل مختلفة (Work Areas).

والجزء الرابع يشرح برنامج حسابات العملاء (Accounts Receivable) الذى يضيف إمكانيات متقدمة للتعامل مع الملفات المرتبطة (Related) وتحديثها واستخدام الملفات التاريخية (History Files) للإحتفاظ بالبيانات القديمة.

والجزء الخامس يضيف بعض الوسائل المتقدمة (Advanced Techniques) التى يمكن لمخطط البرامج استخدامها مع أى برنامج لزيادة كفاءته.

والجزء السادس يشرح بعض التطبيقات الإضافية التى تتضمن تطبيقا منزليا يمكن استخدامه فى تخزين بيانات المعارف والأقارب واسترجاع أو تعديل أو مسح أى بيان. كما يتضمن هذا الجزء أيضا شرح استخدام مولد التطبيقات (Application Generator) الخاص ببرنامج (+ Dbase III).

ويلاحظ أن الكتاب يتدرج فى درجة صعوبة البرامج حتى يصل بالقارىء فى نهاية الكتاب إلى الخبرة الكافية والقدرة على التعامل مع أعقد نظم المعلومات.

ملاحظة

البرامج المشروحة فى هذا الكتاب تعمل على جميع برامج عائلة (DBase) مثل (DBase IV) ، (Clipper) ، (FoxBase +) ، (FoxPro) كما أن الأوامر المستخدمة فى كتابة هذه البرامج هى نفس الأوامر المستخدمة فى برامج عائلة (DBase) الأخرى. إرجع إلى الكتابين السابقين لمراجعة أوامر عائلة (DBase).

١ - ١ مشيرة النقطة (Dot Prompt)

كما سبق الإيضاح فى الكتابين السابقين فإن برنامج (+DBase III) ينقسم إلى جزئين رئيسيين هما برنامج المساعد (Assistant) ومشيرة النقطة (Dot Prompt). واستخدام مشيرة النقطة هو الأساس فى كتابة البرامج.

ويتم عرض مشيرة النقطة عن طريق الضغط على مفتاح الهروب (Esc) عند بدء تشغيل برنامج (+DBase III) وظهور قوائم المساعد (Assistant). ويمكن عرض مشيرة النقطة مباشرة بعد تحميل البرنامج. ولتنفيذ ذلك يتم إجراء تعديل بسيط فى ملف المواصفات (Config.DB) وذلك بالغاء السطرين التاليين من الملف.

STATUS ON
COMMAND = ASSIST

١ - ٢ تعديل مواصفات النظام (Config.sys)

كما سيلاحظ القارىء فيما بعد ، فإن بعض البرامج التى سيتم تصميمها خلال هذا الكتاب تحتوى على العديد من الملفات التى يلزم فتحها فى نفس الوقت ، فإذا كان ملف مواصفات النظام (Config.sys) لايحتوى على الأمر الذى يسمح باستخدام عدد كبير من الملفات فإن من المتوقع عند تشغيل أى برنامج كبير ظهور الرسالة التالية :

Too many files are open

وللتغلب على ذلك يتم تعديل ملف مواصفات النظام (Config.sys) باستخدام برنامج (EDLIN) أو أى برنامج معالجة كلمات مع الشكل غير الوثائقى (Non Document Format). ويتم كتابة السطرين التاليين

FILES = 20
BUFFERS = 15

ويجب بعد ذلك إطفاء الجهاز وإعادة تشغيله من جديد حتى يتم تحميل ملف المواصفات وتخصيص الملفات (Files) ومخازن الذاكرة (Buffers) المطلوبة للبرامج.

1

الجزء الأول

مراجعة كتاب نظم إدارة قواعد البيانات " الجزء الأول"

الفصل الثاني

أنواع البرامج

من المهم قبل البدء فى تصميم البرنامج تحديد نوع هذا البرنامج إذا كان من البرامج التى تتعامل مع ملف قاعدة بيانات واحد (Single Database) مثل برنامج شئون الطلبة (Cadets) كما سيتم الإيضاح ، أو من البرامج التى تتعامل مع عدة ملفات مرتبطة (Related Databases) ، أو من البرامج التى تتعامل مع ملف رئيسى وملفات حركة (Master/Transaction Databases). لذلك سوف يتم إلقاء الضوء على كل نوع من هذه الأنواع الثلاثة قبل تصميم البرامج التى تنتمى إلى كل نوع.

٢ - ١ التعامل مع قاعدة بيانات واحدة

البرنامج فى هذه الحالة يعتبر أبسط أنواع البرامج. حيث يكفى تصميم ملف قاعدة بيانات واحد ويتعامل البرنامج دائما مع هذا الملف. لذلك لاتكون هناك حاجة إلى فتح عدة ملفات واستخدام عدة مناطق عمل (Work Areas) فى نفس الوقت. وهذا يسهل التحكم فى البرنامج ومتابعة تنفيذه.

وعادة يتم إنشاء ملف قاعدة البيانات قبل البدء فى كتابة البرنامج. ويتم ذلك بكتابة السطر التالى من مشيرة النقطة (Dot Prompt).

```
CREATE Cadets
```

حيث يكون الملف (Cadets.dbf) هو ملف قاعدة البيانات المطلوب إنشاؤه. وفى هذه الحالة تظهر قائمة تحديد الحقول التى يتم من خلالها تحديد إسم كل حقل ونوعه وعرضه وعدد الأرقام العشرية إن وجدت.

كما يمكن إنشاء ملف الفهرس من مشيرة النقطة أيضا كالآتى مثلا :

```
USE Cadets
INDEX ON Name TO Name
```

فى هذه الحالة يتم ترتيب السجلات بناء على الترتيب الهجائى لأسماء الطلبة. كما يمكن فهرسة الملفات بناء على حقلين حيث يكون الحقل الأول هو الحقل الرئيسى الذى يتم الترتيب بناء عليه. وذلك كالآتى مثلا :

```
USE Cadets
INDEX ON Name + Class TO Name
```

وعندما يراد فتح ملف الفهرس يتم كتابة الآتى مثلا :

```
USE Cadets INDEX Name
```

وهذا يؤدى إلى فتح ملف قاعدة البيانات وملف الفهرس الخاص به. كما يمكن فتح ملف الفهرس وحده عندما يكون قد سبق فتح ملف قاعدة البيانات وذلك كالآتى :

```
SET INDEX TO Name
```

٢ - ٢ قواعد البيانات المرتبطة (Related Databases)

قواعد البيانات المرتبطة هى الملفات التى تكون مرتبطة ببعضها بناء على حقل مشترك. وهى تفيد بصفة خاصة فى التخلص من أى تكرار للبيانات. وهذا يؤدى إلى تقليل المساحة التخزينية المستخدمة بالإضافة إلى زيادة سرعة التشغيل.

فمثلا عند تصميم ملف قاعدة بيانات لحسابات العملاء (AR.dbf) ، يمكن تصميمه كالآتى :

Structure for database :C: AR.dbf
==================

Field	Field Name	Type	Width	Dec
1	BILL_DATE	Date	8	
2	AMOUNT	Numeric	9	2
3	VENDOR	Character	20	
4	ADDRESS	Character	20	

وهذا يعتبر تصميما سيئا للملف حيث قد يكون هناك بائعون (Vendors) لهم مئات الفواتير بتواريخ مختلفة. وفى هذه الحالة يتم تكرار أسماء هؤلاء البائعين وعناوينهم.

ولكن التصميم الأفضل هو تقسيم الملف إلى ملفين منفصلين ، الملف الأول هو الملف (AR1.dbf) مع تخزين البيانات الشخصية للبائعين مثل الإسم والعنوان فى ملف آخر (AR2.dbf).

فمثلا يمكن إنشاء الملف (AR1) كالآتى :

Structure for database :C: AR1.dbf

Field	Field Name	Type	Width	Dec
1	BILL_DATE	Date	8	
2	AMOUNT	Numeric	9	2
3	VEND_CODE	Character	5	

كما يمكن إنشاء الملف (AR2.dbf) كالآتى :

Structure for database :C: AR1.dbf

Field	Field name	Type	Width	Dec
1	VEND_CODE	Character	5	
2	VENDOR	Character	20	
3	ADDRESS	Character	20	

ويلاحظ أن الحقل الوحيد المكرر هو حقل (Vend_Code). وهو الحقل الذى يستخدم فى ربط الملفين.

ولربط هذين الملفين يتم فهرسة أحدهما على الحقل المشترك (Vend_Code) كالآتى :

```
USE AR2
INDEX ON Vend_Code TO Vendor
```

ثم يتم فتح كل ملف فى منطقة عمل مختلفة (Work Area). ويتم إنشاء العلاقة بين الملفين باستخدام الأمر (SET RELATION TO). وذلك كما يتضح من السطور التالية :

```
SELECT 1
USE AR1
SELECT 2
USE AR2 INDEX Vendor
SELECT 1
SET RELATION TO Vend_Code INTO AR2
```

وعندما يراد عرض أى حقل من حقول الملف (AR2) يستخدم الرمز (<- B). حيث أن (B) هنا يمثل الإسم المرادف (Alias) لمنطقة العمل رقم (2) ، وسيتم شرح ذلك بالتفصيل عند شرح نظام حسابات العملاء (Accounts Receivable).

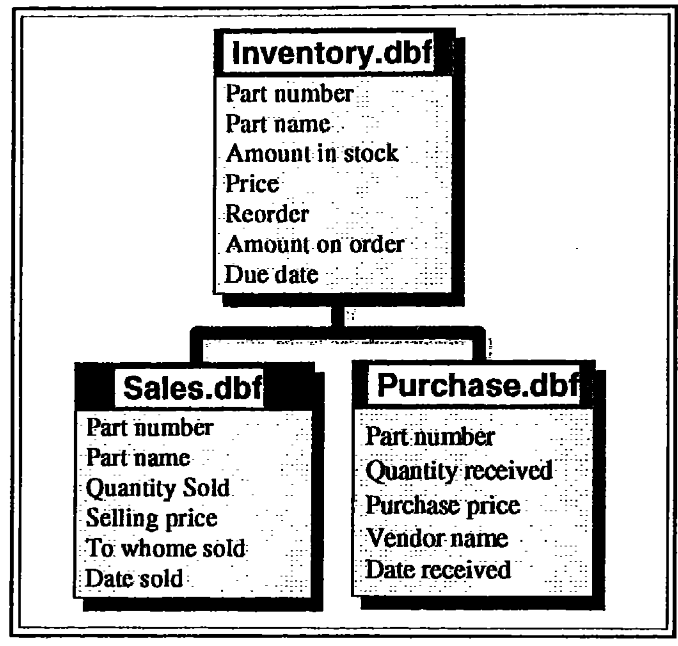
٢ - ٣ نظام الملف الرئيسى وملفات الحركة

هذا النوع من البرامج يستخدم عادة مع نظم المخازن (Inventory) والمكتبات (Libraries) والبنوك (Banks) حيث يكون هناك ملف بيانات رئيسى (Master File) يحتوى على بيانات كل حركة إضافة أو صرف. ويتم تحديث بيانات الملف الرئيسى (Master) من ملفات الحركة (Transaction). فمثلا نظام المخازن فى أبسط صورة يحتوى على ملف رئيسى واحد وملفين حركة. ويكون هيكل النظام كما هو موضح من الشكل (٢ - ١).

والملف الرئيسى فى هذا النظام مثلا يحتوى على البيانات الحالية عن المخزون الفعلى (On hand stock). كما يحصل على بيانات كل حركة صرف أو توريد من ملفات الحركة حتى يعطى دائما الموقف الحالى للمخزون. وفائدة هذا النظام أنه يعطى المستخدم الموقف الحالى فى كل لحظة كما يتابع حركة الصرف والتوريد.

ويلاحظ وجود حقل مشترك بين الملف الرئيسى وملفات الحركة وهو حقل رقم الجزء (Part_No) وهو حقل منفرد (Unique) يستخدم فى ربط الملفات الثلاثة. ويستخدم الأمر (UPDATE) فى تحديث الملف الرئيسى من ملفات الحركة. ولتنفيذ ذلك يجب

أولا فهرسة الملفات على حقل رقم الجزء (Part_No).



شكل (٢ - ١)

فمثلا لتحديث الملف الرئيسى (Master) من ملف المبيعات (Sales) يجب أولا كتابة السطور التالية :

```
USE Inventory
INDEX ON Part_No TO Master
USE Sales
INDEX ON Part_No TO Sales
```

ثم يتم فتح كل ملف فى منطقة عمل مختلفة (Work Area) باستخدام الأمر (SELECT) ، ثم يستخدم الأمر (UPDATE) فى نقل البيانات من ملف المبيعات إلى الملف الرئيسى. وذلك كالآتى :

```
SELECT 2
USE Sales INDEX Sales
SELECT 1
USE Inventory INDEX Master
UPDATE ON Part_No FROM Sales REPLACE On_Hand ;
WITH  On_Hand - B -> Qty
```

ملاحـظة

يجب ملاحظة أنه عند زيادة الأمر عن طول السطر على الشاشة يتم استخدام حرف الفاصلة المنقوطة (;). ويجب أن يعرف مخطط البرامج وظيفة الفاصلة المنقوطة فى هذه الحالة حتى لايحدث خطأ فى كتابة الأمر حيث أن هذا الحرف يؤدى إلى ضم السطر التالى إلى السطر الجارى كتابته. ولذلك يراعى عندما يكون مطلوبا وجود مسافة خالية (Space) ترك هذه المسافة قبل كتابة هذا الحرف لأن هذا الحرف يضم السطر التالى دون أى مسافات. ويلاحظ هذا فى السطر الأخير من الأوامر السابقة. وهذا السطر يؤدى إلى تحديث الملف الرئيسى (Inventory.dbf) من ملف المبيعات (Sales.dbf) عن طريق استبدال محتويات حقل الكمية الفعلية (On_Hand) بنفس المحتويات مطروحا منها كمية الصنف التى تم بيعها (B -> Qty) والتى يتم الحصول عليها من ملف المبيعات فى منطقة العمل رقم (٢).

ويمكن إجراء عملية التحديث من ملف المشتريات (Purchases) بنفس الطريقة. والفارق الوحيد هو إضافة الكمية المشتراة (B -> Qty) من ملف المشتريات فى منطقة العمل رقم (٢) إلى الكمية الفعلية. وذلك كالآتى :

```
UPDATE ON Part_No FROM Purchase REPLACE On_Hand WITH ;
On_Hand + B -> Qty
```

وسوف يتم شرح كل الوسائل المتاحة لكتابة البرامج الخاصة بهذا النظام فى الجزء الخاص بدراسة برنامج المخازن (Inventory).

٢ - ٤ استخدام حقول الملاحظات

هناك بعض قواعد البيانات التى تحتاج إلى استخدام حقل للملاحظات. وأوضح مثل لذلك قاعدة بيانات المكتبة (Library). حيث يراد مثلا إنشاء ملف يحتوى على بيانات بالأبحاث أو الكتب الخاصة بكل مؤلف. هذا الملف يمكن أن يحتوى على حقول إسم المؤلف، عنوان البحث أو الكتاب ، تاريخ النشر ، إسم الناشر ، الموضوعات (Topics) ، والملخص (Abstract). ويمكن تصميم هيكل الملف (Structure) فى هذه الحالة بطريقتين. فى الطريقة الأولى نسمى الملف مثلا (Lib1.dbf) وندخل الحقول الخاصة به كالآتى :

Structure for database :C: Lib1.dbf				
Field	Field name	Type	Width	Dec
1	AUTHOR	Character	20	
2	TITLE	Character	20	
3	PUB	Character	20	
4	DATE	Date	8	
5	TOPICS	Character	60	
6	ABSTRACT	Character	254	

شكل (٢ - ٢)

ويلاحظ فى هذه الحالة أنه تم تخصيص (٢٥٤) حرفا للملخص (Abstract). وذلك لأن هذا العدد يمثل الحد الأقصى لعرض الحقل الحرفى. وهذا العرض يسمح بكتابة سطور معدودة لاتزيد عن أربعة سطور.

ولكن قد يكون مطلوبا كتابة ملخص كبير يزيد عن صفحة لكل كتاب مثلا. وفى هذه الحالة يتم استخدام نوع آخر من الحقول يسمى حقل الملاحظات (memo field). وهذا النوع من الحقول يسمح بكتابة حتى (٤٠٠٠) حرفا فى الحقل الواحد. ويمكن زيادة عدد الحروف أكثر من ذلك عن طريق استخدام معالج كلمات (Word Processor) آخر غير المستخدم مع برنامج (DBase III+).

والشكل (٢ - ٣) يوضح هيكل الملف (Lib2.dbf) بعد استخدام حقل الملاحظات (memo field). ويلاحظ من الشكل أن البرنامج يخصص الرقم (١٠) آليا لعرض حقل الملاحظات (Abstract) رغم أنه يسمح فعليا بتخزين حتى (٤٠٠٠) حرف. وهذا لأن ما يكتب فى هذا الحقل يخزن فعليا فى ملف قاعدة بيانات مساعد (Auxiliary) يكون امتداده (dbt.) وليس (dbf.). ولكن مكان هذا الحقل فقط هو الذى يتم تخزينه فى ملف قاعدة البيانات الأصلى.

Structure for database Lib2.dbf				
Field	Field name	Type	Width	Dec
1	AUTHOR	Character	20	
2	TILTE	Character	20	
3	PUB	Character	20	
4	DATE	Date	8	
5	TOPICS	Character	60	
6	ABSTRACT	Memo	10	

شكل (٢ - ٣)

ولإدخال بيانات فى حقل الملاحظات يتم ذلك من خلال شاشة الإدخال حيث يتم وضع المؤشر على حقل الملاحظات والضغط على مفتاحى (Ctrl-PgDn). وفى هذه الحالة يتم مسح الشاشة وتصبح الشاشة جاهزة لإدخال بيانات هذا الحقل. وعند الإنتهاء من الكتابة يتم الضغط على مفتاحى (Ctrl-PgUp) أو مفتاحى (Ctrl-w) أو مفتاحى (Ctrl-end) لتخزين بيانات هذا الحقل والرجوع إلى شاشة الإدخال مرة ثانية.

ويلاحظ عند استخدام الأمر (LIST) فى عرض بيانات الحقول عدم ظهور محتويات حقل الملاحظات ولكن تظهر كلمة (memo). ولكن عندما يراد عرض محتويات هذا الحقل يتم تحديد إسم الحقل مع الأمر (LIST) كالآتى مثلا :

```
LIST OFF Author,Title,Pub,Date,Abstract
```

وفى هذه الحالة يظهر الموضح بالشكل (٢ - ٤)

ويمكن التحكم فى عرض الكتابة فى حقل الملاحظات عن طريق كتابة الأمر (SET MEMOWIDTH). فمثلا لتحديد العرض (٤٠) لحقل الملاحظات يتم كتابة الأمر التالى :

```
SET MEMOWIDTH TO 40
```

Garton J. T. Decision Support Systems Byte Magazine 03/01/85
This article discusses automated Decissions , Support Systems used in modern business Mini and Microcomputer systems. It includes a review and comparison of several currently available systems , both as support , knowledge Maker , Mind games and Decisive.

Franklin B.W. Automated MBO MBO Monthly 05/01/85
Describes several automated systems that support Management by objectives (MBO). The basic theory of MBO is discussed, the several MBO systems are reviewed.

شكل (٢ - ٤)

كما يمكن التحكم فى مكان ظهور الملاحظات وشكلها من خلال ملف الأوامر (Command File). فمثلا لعرض نفس الملاحظات السابقة بطريقة أوضح يمكن كتابة البرنامج التالى :

```
* * * * * * * * * * * * * * * * * * * * * * * * * *Library.prg
* - - Sample Program to print database with memo field
USE Lib2
GO TOP
DO WHILE .NOT. EOF()
   ? "AUTHOR     :", Author
   ? "Title      :", Title
   ? "Publisher  :", Pub
   ? "Date       :", Date
   ? "Keywords   :", Topics
   ?
   ? Abstract
   ?
   ?
   SKIP
ENDDO(while not eof)
```

وعند تنفيذ هذا البرنامج تظهر البيانات الواضحة فى الشكل (٢ - ٥).

Record no. : 1
Author : Garton , J ,
Title : Decission Support Systems
Publisher : Byte Magazine
Date : 03/01/89
Key words : Support,knowledge Maker , Mind games

This article discusses automated Decission Support sytems used in modern business. Mini and micro-------------------
----

Record no. : 2
Author : Franklin B. W.
Title : Automated MBO
Publisher : MBO Monthly
Date : 05/01/89
Key.words : Management , MBO , Microcomputer

Describes several automated systems---------------------

شكل (٢ - ٥)

وهناك قصور واحد في استخدام حقول الملاحظات وهو أنه لايمكن البحث عن الملاحظات التي تحتوى على كلمة معينة مثلا أو موضوع معين. فمثلا عند استخدام الأمر التالي :

LIST FOR "Computer" $ Abstract

للبحث عن السجلات التي تحتوى على كلمة (Computer) في حقل الملاحظات (Abstract) في هذه الحالة يلاحظ ظهور الرسالة التالية :

Operation with memo field invalid

ولعلاج هذه المشكلة يمكن استخدام حقل آخر للبحث يسمى (Topics) كما في المثال السابق حيث يتم وضع بعض رؤوس المواضيع (Topics) التي يمكن بعد ذلك البحث من خلالها.

فمثلا عند كتابة الأمر التالي :

LIST FOR "Computer" $ Topics

يتم عرض بيانات جميع السجلات التي تحتوى على كلمة (Computer) في حقل الملاحظات.

الفصل الثالث

البحث السريع

هذا الفصل يركز على الوسائل المختلفة (Techniques) التى يمكن استخدامها فى كتابة البرامج لزيادة سرعة التشغيل بدرجة كبيرة. ومن أهم العمليات التى تؤثر بدرجة كبيرة فى سرعة تنفيذ البرنامج عملية البحث عن بيان معين خلال ملف قاعدة البيانات حيث أن ذلك قد يستغرق أياما فى الملفات الكبيرة إذا لم يتم كتابة البرامج بالصورة السليمة. أما عند استخدام الوسائل المختلفة التى سيتم شرحها فى هذا الفصل فقد يصل زمن البحث إلى دقائق معدودة وربما ثوان مهما كبر حجم ملف قاعدة البيانات. وأول هذه الوسائل هو استخدام الفهرس (Index) فى ترتيب سجلات الملف.

٣ - ١ استخدام الفهرس

أقرب وأوضح مثال لتأثير الفهرس على سرعة البحث عن البيانات هو إستخدام فهرس الكتاب. نفرض مثلا أننا نقرأ فى كتاب عن الحاسبات ونريد أن نبحث عن موضوع نظم الخبرة فهناك طريقتان للبحث ، الأولى عن طريق فرز صفحات الكتاب صفحة صفحة حتى نصل إلى الصفحة التى تحتوى على هذا الموضوع وإذا كان الكتاب كبيرا فإن البحث قد يستغرق مدة طويلة. والطريقة الثانية هى الذهاب مباشرة إلى الفهرس الموجود فى آخر الكتاب والبحث فى الفهرس عن كلمة (نظم الخبرة) وتحديد رقم الصفحة المناظر ثم الذهاب إلى هذه الصفحة. وعملية البحث فى فهرس الكتاب لن تأخذ وقتا كبيرا لأن هذا الفهرس يكون مرتبا بالترتيب الهجائى للحروف.

وما يحدث مع برامج عائلة (DBase) هو نفس الشىء تقريبا حيث تكون هناك طريقتان للبحث عن بيان معين فى ملف قاعدة البيانات. الطريقة الأولى عن طريق قراءة كل سجل من سجلات الملف للوصول إلى السجل الذى يحتوى على البيان المطلوب البحث عنه. والطريقة الثانية هى إنشاء ملف فهرس (Index) بناء على الحقل الذى يحتوى على البيان المطلوب البحث عنه كالإسم مثلا والبحث عن البيان المطلوب خلال هذا الفهرس وتحديد رقم السجل المقابل له ثم الذهاب إلى هذا السجل.

وفى برامج عائلة (DBase) يتم إنشاء الفهرس باستخدام الأمر (INDEX ON). كما يتم تشغيل هذا الفهرس بكتابة إسمه عند فتح ملف قاعدة البيانات كالآتى مثلا :

```
USE Cadets INDEX Name
```

كما يتم استخدام الأمر(FIND) أو الأمر(SEEK) فى البحث عن البيان المطلوب.

وعند إجراء أى عمليات على سجلات ملف قاعدة البيانات مثل إضافة سجلات جديدة أو مسح سجلات أو تعديل بيانات سجل معين يجب مراعاة فتح ملف الفهرس أولا قبل إجراء هذه العمليات حيث أن فتح ملف الفهرس يؤدى إلى إدخال أى تعديل يتم على ملف قاعدة البيانات على هذا الفهرس.

٣ - ٢ طـرق البحـث

بالإضافة إلى السرعة التى يوفرها استخدام الفهرس فى ترتيب السجلات فإن هناك طرقا مختلفة للبحث عن السجلات توفرها برامج عائلة (DBase) وتتفاوت سرعة البحث من خلالها. لذلك فمن المهم عرض طرق البحث المختلفة ومقارنة سرعة البحث فى كل طريقة حتى يختار مخطط البرامج الطريقة المناسبة التى تزيد من كفاءة وسرعة تنفيذ البرنامج.

فمثلا نفرض أن هناك ملف قاعدة بيانات إسمه (Test.dbf) يحتوى على ألف سجل. ونفرض أن هناك عشرة سجلات تحتوى على الإسم (Mohamed) مثلا. ويراد عرض بيانات هذه السجلات على الشاشة. فى هذه الحالة نقوم بمقارنة طريقتين مختلفتين لتنفيذ المطلوب.

فى الطريقة الأولى يتم استخدام الأمر (LIST) مع كلمة (FOR) لإدخال شرط البحث. والسطور التالية توضح ذلك :

```
CLEAR
USE Test INDEX Name
ACCEPT "List what name ? " TO Search
LIST FOR Name = Search
```

وعند تنفيذ هذا البرنامج يتم مسح الشاشة ويظهر السؤال التالى :

```
List what name ?
```

وعند إدخال الإسم (Mohamed) مثلا يتم تخزينه فى متغير الذاكرة الحرفى (Search) وتظهر بيانات السجلات العشرة التى تحتوى على هذا الإسم. والوقت الذى

يستهلك فى عرض هذه السجلات يزيد عن الدقيقتين بقليل. وذلك فى حالة استخدام الأقراص المرنة (Floppy Disks). كما يستهلك حوالى ٣٢ ثانية عند استخدام القرص الصلب (Hard Disk).

والطريقة الثانية لتنفيذ نفس هذه العملية هى استخدام الأمر (FIND) أو الأمر (SEEK) فى تحديد رقم أول سجل يحتوى على الإسم (Mohamed) ثم استخدام (WHILE) لعرض باقى السجلات. والسطور التالية توضح ذلك :

```
CLEAR
USE Test INDEX Name
ACCEPT "List what name? " TO Search
SEEK Search
LIST WHILE Name = Search
```

وفى هذه الحالة يتم عرض بيانات السجلات العشرة فى خمس ثوان فى حالة استخدام الأقراص المرنة (Floppy Disks) ويتم ذلك فى أربع ثوان فى حالة استخدام القرص الصلب (Hard Disk).

٣ - ٣ البحث عن مدى معين

عند البحث عن مدى معين من السجلات محصور بين قيمة معينة لأحد الحقول وقيمة أخرى لهذا الحقل فإن ذلك يمكن أن يتم باستخدام الأمر (LIST) مع كلمة (FOR). كما يمكن استخدام الفهرس مع استخدام الأمر (SEEK) و (WHILE) لتنفيذ نفس العملية. ولتوضيح الفرق بين الحالتين سنقوم بدراسة المثال التالى :

نفرض أنه يراد البحث عن السجلات التى تبدأ من تاريخ معين فى حقل التاريخ (Date) وتنتهى بتاريخ آخر. فى هذه الحالة نبدأ باستخدام الأمر (LIST) مع كلمة (FOR) كالآتى :

```
USE Test
CLEAR
STORE "    " TO Start , Finish
@ 10,2 SAY "Enter start date" GET Start ;
```

```
    PICT "99/99/99"
@ 12,2 SAY "Enter ending date" GET Finish ;
    PICT "99/99/99"
READ
STORE CTOD(Start) TO Start
STORE CTOD(Finish) TO Finish
LIST FOR Date > = Start .AND. Date < = Finish
```

فى هذه الحالة يتم عرض بيانات السجلات المحصورة بين التاريخين الذين يكتبهما المستخدم.

وعند استخدام الطريقة الأخرى يتم كتابة السطور التالية :

```
USE Test INDEX Dates
CLEAR
STORE "      " TO Start , Finish
@ 10,2 SAY "Enter start date";
  GET Start PICT "99/99/99"
@ 12,2 SAY "Enter ending date";
  GET Finish PICT "99/99/99"
READ
STORE CTOD(Start) TO Start
STORE CTOD(Finish) TO Finish
SEEK Start
LIST WHILE Date < = Finish
```

وفى هذه الحالة يتم عرض السجلات أسرع كثيرا من الطريقة الأولى ، حيث أن البرنامج يبحث أولا عن السجل المحتوى على تاريخ البداية (Start). وحيث أن الملف مفهرس على حقل التاريخ (Dates) ، فإن البحث يتم عن السجلات التى تبدأ من السجل الذى تم تحديده بواسطة الأمر (SEEK) وتنتهى بالتاريخ الموجود فى المتغير (Finish). ومع أن هذه الطريقة أسرع كثيرا من الطريقة الأولى إلا أنها تنطوى على شىء من الخطورة. حيث أن الأمر (SEEK) إذا لم يجد السجل المحتوى على تاريخ البداية (Start) فإن

البرنامج لايعرض أى سجلات. لأن مؤشر السجلات (Record Pointer) ينتقل إلى آخر الملف (End of File). ولذلك يفضل فى هذه الحالة استخدام طريقة أخرى تجمع بين الطريقتين للإحتفاظ بسرعة تنفيذ البرنامج. ويتضح ذلك من السطور التالية :

```
USE Test INDEX Dates
CLEAR
STORE "      " TO Start , Finish
@ 10,2 SAY "Enter start date" ;
  GET Start PICT "99/99/99"
@ 12,2 SAY "Enter ending date" ;
  GET Finish PICT "99/99/99"
READ
STORE CTOD(Start) TO Start
STORE CTOD(Finish) TO Finish
SEEK Start
IF FOUND()
   LIST WHILE Date < = Finish
ELSE
   LIST FOR Date > = Satart .AND. Date < = Finish
ENDIF(not found)
```

وهناك طريقة أخرى لتنفيذ نفس المطلوب عن طريق استخدام الأمر (SET FILTER TO) وذلك كالآتى مثلا :

```
SET FILTER TO DATE > = Start .AND. DATE < = Finish
LIST
```

٣ - ٤ التنفيذ السريع للعمليات الحسابية

هناك بعض الأوامر الحسابية مثل (COUNT) ، (SUM) ، (AVERAGE). وهذه الأوامر يمكن أيضا تقليل وقت تنفيذها للعملية الحسابية بدرجة كبيرة. فمثلا لحساب عدد السجلات التى تحتوى على الإسم (Mahmoud) يمكن كتابة السطرين التاليين :

```
USE Test
COUNT FOR Name = "Mahmoud"
```

وهذه الطريقة تستهلك حوالى (١٥) ثانية للوصول إلى النتيجة المطلوبة وهى وجود عشرة سجلات تحتوى على هذا الإسم.

ويمكن تقليل هذا الوقت بدرجة كبيرة جدا بكتابة الأوامر التالية :

```
USE Test INDEX Name
FIND Mahmoud
COUNT WHILE Name = "Mahmoud"
```

فى هذه الحالة يتم تنفيذ المطلوب فى حوالى ثانية واحدة. ونفس هذه الطريقة يمكن استخدامها مع الأمر (SUM) والأمر (AVERAGE).

٣ - ٥ زيادة سرعة طباعة التقارير

يمكن استخدام طريقتين أيضا فى طباعة التقارير. ولتوضيح الفرق بينهما يتم كتابة السطرين التاليين الذين يمثلان الطريقة الأولى.

```
USE Test INDEX Name
REPORT FORM Rep1 FOR Name ="Mahmoud"
```

وهذه الطريقة تستغرق حوالى ٣٠ ثانية فى عرض التقرير على الشاشة. والسطور التالية توضح الطريقة الثانية :

```
USE Test INDEX Name
FIND Mahmoud
REPORT FORM Rep1 WHILE  Name = "Mahmoud"
```

وهذه الطريقة تستغرق حوالى ٦ ثوان فى عرض التقرير على الشاشة.

٣ - ٦ النسخ السريع للسجلات

عندما يراد نسخ مجموعة من السجلات فى ملف مؤقت (Temporary) ، يمكن تنفيذ ذلك بطريقتين :

الطريقة الأولى يتم توضيحها من السطرين التاليين :

```
USE Test INDEX Name
COPY TO Temp FOR Name = "Mahmoud"
```

وهذه العملية تستهلك حوالى ٣٠ ثانية على القرص الصلب (Hard Disk).

والطريقة الثانية يتم توضيحها من السطور التالية :

```
USE Test INDEX Name
FIND Mahmoud
COPY TO Temp WHILE Name = "Mahmoud"
```

هذه الطريقة تؤدى إلى تقليل زمن التنفيذ إلى مايقرب من ثانيتين.

٣ - ٧ التعامل مع ملفات فهرس متعددة

يحتاج مخطط البرامج إلى ترتيب السجلات ترتيبا مختلفا حسب العملية المطلوب إجراؤها وفى هذه الحالة يمكنه استخدام عدة ملفات فهرس. وبرامج عائلة (DBase) تتيح لمخطط البرامج إنشاء أى عدد من ملفات الفهرس ولكنه لايستطيع فتح أكثر من سبعة ملفات فهرس فى نفس الوقت.

فمثلا فى برنامج شئون الطلبة (Cadets) يمكن إنشاء فهرس بناء على حقل الإسم (Name) كالآتى :

```
USE Cadets
INDEX ON Name TO Name
```

كما يمكن إنشاء فهرس بناء على رقم الفرقة مثلا (Class) كالآتى :

```
USE Cadets
INDEX ON CLass TO Class
```

ويمكن فتح الفهرسين معا بكتابة السطر التالى :

```
USE Cadets INDEX Name , Class
```

وترتيب كتابة ملفات الفهرس مهم جدا فى هذه الحالة حيث أن الملف الأول (Name) يصبح الفهرس الرئيسى (Primary). فعند استخدام أى أمر من أوامر التعامل مع السجلات مثل (LIST) ، (REPORT) فإن البرنامج يعرض السجلات مرتبة حسب الفهرس الرئيسى. وإذا تساوت بيانات بعض السجلات فى حقل الفهرس الرئيسى يتم ترتيبها بناء على الفهرس الثاني وهكذا. كما أن الأوامر (SEEK) ، (FIND) تبحث خلال الفهرس الرئيسى فقط.

وإذا أريد تعديل ترتيب ملفات الفهرس يمكن كتابة السطر التالى مثلا :

```
USE Cadets INDEX Class , Name
```

فى هذه الحالة يصبح الملف الأول (Class.ndx) هو الفهرس الرئيسى. ويراعى دائما فتح جميع ملفات الفهرس التى سبق إنشاؤها ثم تعديل ترتيبها بعد ذلك حسب الحاجة. وذلك لأن أى تعديل يحدث فى السجلات مثل إضافة سجلات جديدة أو مسح سجلات أو تعديل سجلات يؤدى إلى تحديث ملفات الفهرس المفتوحة. أما ملفات الفهرس غير المفتوحة فلا يتم تحديثها وبالتالى تصبح غير مطابقة للوضع الحالى للسجلات فى ملف قاعدة البيانات. ولعلاج ذلك يتم إعادة إنشاء الفهرس من جديد. ويتم ذلك عن طريق فتح ملفات الفهرس التى سبق إنشاؤها ثم استخدام الأمر (REINDEX) وذلك كالآتى :

```
USE Cadets INDEX Name , Class
REINDEX
```

الفصل الرابع

خطوات تصميم النظام

عادة يبدأ تصميم النظام بمجرد فكرة ثم تنمو هذه الفكرة تدريجيا حتى تنتج النظام الكامل. وعادة يبدأ مخطط البرنامج بأن يسأل نفسه (من أين أبدأ ؟). وعندما يجيب على هذا السؤال ويبدأ فى التنفيذ فإنه يسأل نفسه بعد كل خطوة (أين أذهب بعد هذه الخطوة). ومن مجموع هذه الخطوات يصل فى النهاية إلى التصميم النهائى للنظام.

وخطوات التصميم بصفة عامة يمكن شرحها كالآتى :

١ - تعريف المشكلة أو الهدف من النظام.
٢ - توصيف المدخلات والمخرجات.
٣ - تصميم هيكل قاعدة البيانات (Database Structure).
٤ - تقسيم البرنامج إلى برامج فرعية (Modules) يؤدى كل منها وظيفة محددة.
٥ - كتابة البرامج الفرعية.
٦ - اختبار وتصحيح البرنامج.

وسوف يتم شرح كل خطوة من هذه الخطوات فى هذا الفصل.

٤ - ١ تعريف المشكلة (Problem Definition)

أول خطوة فى تصميم النظام هى تحديد المشكلة المطلوب حلها أو الهدف العام للنظام. وكلما كان هذا الهدف محددا وواضحا كان تنفيذ الخطوات التالية أسهل. وأول خطوة فى هذا التحديد تبدأ من إسم النظام نفسه مثل (نظام معلومات شئون الطلبة). وهذا يعنى أن المطلوب إنشاء نظام يتيح الحصول على معلومات معينة عن الطلبة. ولكن هذا الإسم وحده يكون غامضا بعض الشىء. لذلك يتم توضيحه قليلا عن طريق معرفة خصائص المستخدم النهائى لهذا النظام. فإذا كان المطلوب إنشاء نظام معلومات يتيح للمستخدم العادى (الذى ليس له أى خبرة بالحاسب) استخدام النظام وإسترجاع المعلومات المطلوبة، فى هذه الحالة يصبح من السهل على مخطط البرامج تحليل هذا الهدف إلى خطوات محددة مثل الآتى :

١ - إضافة بيانات طلبة جدد.
٢ - طباعة تقارير متضمنة بيانات طالب معين أو مجموعة من الطلبة.
٣ - تعديل بيانات أى طالب.
٤ - مسح بيانات أى طالب.

كما يمكن تحليل هذه الخطوات إلى خطوات أخرى أكثر تحديدا.

٤ - ٢ توصيف المدخلات والمخرجات (Input/Output)

الخطوة الثانية فى تصميم النظام هى تحديد مايجب إدخاله إلى الحاسب (Input) ومايجب أن نحصل عليه منه (Output). ولايهم فى هذه المرحلة كيفية الإدخال أو الإخراج. فمثلا فى نظام معلومات شئون الطلبة (Cadets) نريد الحصول على الآتى من الحاسب.

١ - تقارير بيانات الطلبة (Reports) تحتوى على الإسم والعنوان ورقم التليفون ... الخ.
٢ - تقارير مختصرة للطلبة (Labels) تحتوى على الإسم والعنوان.

وعادة يتم تحديد المدخلات عن طريق هذه المخرجات. لذلك نستطيع تحديد المدخلات الآتية :

Name
Nationality
Address
Phone number

----

٤ - ٣ تصميم قاعدة البيانات

الخطوة الثالثة هى تصميم قاعدة البيانات المطلوبة. وفى هذه المرحلة يجب تحديد نوع كل حقل من حقول ملف قاعدة البيانات. وهنا يجب التمييز بين الحقول العددية (Numeric) وبين الحقول الحرفية (Character) التى تحتوى على أعداد. حيث أن المقصود بالحقول العددية عادة هو الأعداد التى لايمكن استخدامها فى عمليات حسابية فمثلا رقم التليفون يتم إدخاله كحقل حرفى وليس عدديا.

ويجب فى هذه المرحلة أيضا إنشاء ملف الفهرس. ويجب تحديد نوع هذا الملف حسب الحقل المطلوب البحث بناء عليه. فمثلا فى نظام معلومات شئون الطلبة يتم كتابة السطرين التاليين لإنشاء ملف الفهرس :

```
USE Cadets
INDEX ON Name TO Name
```

٤ - ٤ التصميم المجزأ للنظام (Modular Design)

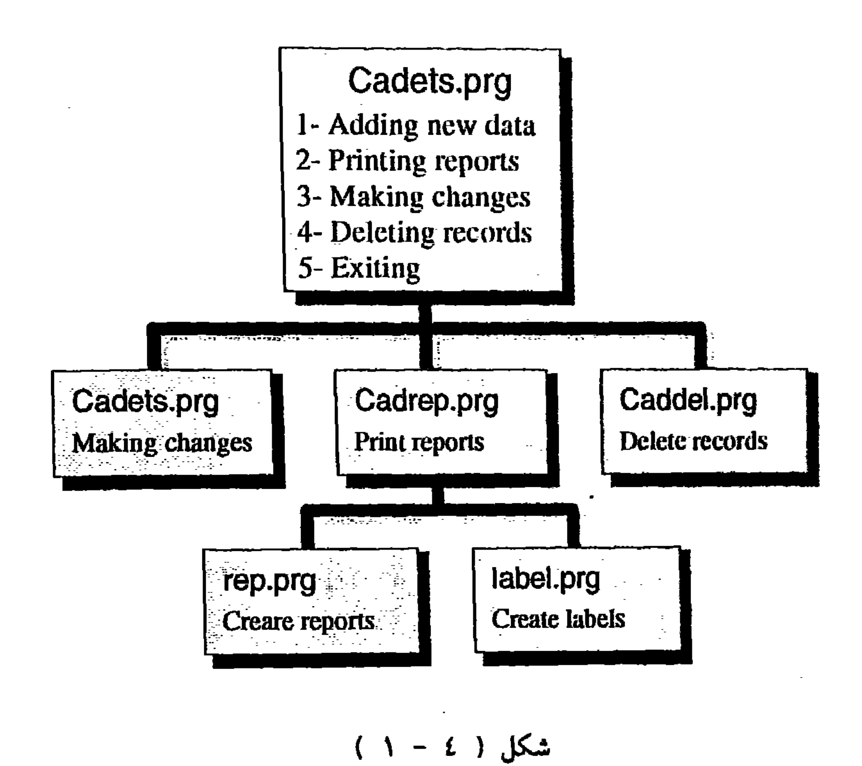
كما سبق الإيضاح فإن أسهل طريقة لتصميم النظام هى تقسيمه إلى برامج صغيرة (Modules) كل منها يحقق وظيفة محددة.

ويستخدم الشكل الهرمى (Hierarchical) فى تحديد البرنامج الرئيسى والبرامج الفرعية المتفرعة منه. فمثلا فى نظام معلومات شئون الطلبة (Cadets) يكون البرنامج الرئيسى هو البرنامج الذى يقوم بعرض القائمة الرئيسية (Main Menu) والبرامج الفرعية هى التى تحقق كل اختيار من اختيارات القائمة. والبرنامج الرئيسى مثلا يعرض القائمة التالية :

```
Cadets Information System
==============
1 - Add new names and addresses
2 - Print reports
3 - Make changes
4 - Delete names and addresses
5 - Exit

Enter choice:
```

والشكل التالى يوضح التركيب الهرمى للبرنامج :


Cadets.prg
1- Adding new data
2- Printing reports
3- Making changes
4- Deleting records
5- Exiting
Cadets.prg
Making changes
Cadrep.prg
Print reports
Caddel.prg
Delete records
rep.prg
Creare reports
label.prg
Create labels

شكل (٤ - ١)

الفصل الخامس

كتابة البرامج

يتم كتابة البرامج من خلال برنامج (+DBase III) أو برامج عائلة (DBase) الأخرى عن طريق كتابة ملفات الأوامر (Command Files) التى تحتوى على الأوامر المطلوب تنفيذها بالتسلسل المنطقى المطلوب.

ويستخدم الأمر (MODIFY COMMAND) فى كتابة ملفات الأوامر. حيث أن هذا الأمر يؤدى إلى تشغيل المصحح الخطى (Text Editor) الخاص ببرنامج (+DBase III) الذى يتم عن طريقه كتابة ملفات الأوامر وتصحيحها. كما يستخدم الأمر (DO) فى تشغيل ملفات الأوامر.

٥ - ١ إنشـــاء ملفـات الأوامـــر

كما سبق الإيضاح فإن لبرنامج (+DBase III) المصحح الخطى الخاص به. وهذا المصحح الخطى يتم تشغيله عن طريق الأمر (MODIFY COMMAND). فمثلا لإنشاء الملف (Test) يتم كتابة الأمر التالى :

```
MODIFY COMMAND Test
```

وعند الضغط على مفتاح الإدخال تظهر شاشة خالية لكتابة الأوامر خلالها. وتظهر قائمة مساعدة (Help) توضح المفاتيح التى يتم عن طريقها التحكم فى الكتابة. وهذه المفاتيح يتم شرح وظائفها من خلال الجدول الموضح بالشكل (٥ - ١).

ويمكن كتابة سطرين من الملف (Test) للتمرين على كتابة ملفات الأوامر كالآتى مثلا :

```
CLEAR
? "Good morning"
```

ويتم الضغط على مفتاح الإدخال بعد كتابة كل سطر. وإذا حدث خطأ فى الكتابة يتم استخدام مفاتيح التصحيح الموجودة فى الجدول فى الوصول إلى الحروف المطلوب تعديلها. ثم يتم تخزين الملف عن طريق كتابة (End ^) أو كتابة (W^). وهذا يؤدى إلى تخزين الملف (Test) والعودة إلى مشيرة النقطة.

المفتاح	الوظيفة
↑ أو E	تحريك المؤشر سطرا لأعلى
↓ أو X	تحريك المؤشر سطرا لأسفل
S أو <-	تحريك المؤشر حرفا لليسار
^D أو ->	تحريك المؤشر حرفا لليمين
Del أو G	مسح الحرف فوق المؤشر
Ins أو ^V	التحويل إلى حالة الإضافة (Ins) أو الكتابة الفوقية.
End أو ^F	تحريك المؤشر كلمة الى اليمين
Home أو ^A	تحريك المؤشر كلمة الى اليسار
^N	إضافة سطر خال مكان المؤشر
^T	مسح كلمة يمين المؤشر
^Y	مسح سطر مكان المؤشر
^End أو ^W	تخزين ملف الأوامر
Esc أو ^Q	الرجوع إلى مشيرة النقطة دون تخزين الملف
^Ka	يعيد تشكيل النص (يستخدم عادة مع حقول الملاحظات)
^KF	البحث عن كلمة معينة فى النص
^KL	تحديد مكان الكلمة الثانية التى يتم البحث عنها
^KR	قراءة ملف خارجى فى مكان المؤشر
^KW	كتابة ملف معين فى ملف آخر باسم آخر

شكل (٥ - ١)

والبرنامج يضيف الإمتداد (prg) لإسم الملف وعندما يراد تشغيل هذا الملف يتم كتابة الآتى :

DO Test

وعند الضغط على مفتاح الإدخال يلاحظ مسح الشاشة وظهور الآتى :

Good morning

ويلاحظ أن البرنامج (Test.prg) قام بتنفيذ السطرين. حيث بدأ بمسح الشاشة (Clear) ثم عرض السطر السابق.

وعندما يراد تعديل هذا البرنامج مثلا يتم كتابة السطر التالى :

MODIFY COMMAND Test

وفى هذه الحالة تظهر السطور السابق كتابتها ويتم تعديلها.

٥ - ٢ التفاعل مع المستخدم

فى المثال السابق كان المطلوب فقط عرض رسالة على الشاشة. ولكن فى معظم البرامج يكون مطلوبا سؤال المستخدم وانتظار إجابته ثم تخزين هذه الإجابة فى متغير ذاكرة وبناء على قيمة هذا المتغير يتم تنفيذ عملية معينة. وهذا يشبه الحديث بين شخصين وتصرف كل منها بناء على ذلك. وهناك عدة أوامر يتم استخدامها فى تحقيق هذا التفاعل (Interaction) بين المستخدم والحاسب. وهذه الأوامر يتم إلقاء الضوء عليها فى الأجزاء التالية.

٥ - ٢ - ١ الأمر (ACCEPT)

يستخدم هذا الأمر فى عرض رسالة للمستخدم وانتظار إجابته على هذه الرسالة ثم تخزين هذه الإجابة فى متغير ذاكرة حرفى. فمثلا يمكن كتابة السطر التالى :

ACCEPT "Send report to printer ? (Y/N)" TO Pr

وعند الضغط على مفتاح الإدخال تظهر الرسالة التالية على الشاشة :

Send report to printer ? (Y/N)

وينتظر البرنامج حتى يدخل المستخدم الإجابة ثم يقوم بتخزين هذه الإجابة فى المتغير (Pr). وهذا الأمر يفضل استخدامه عندما تكون الإجابة المنتظرة من المستخدم حرفية وليست عددية. وهذا لايعنى أنه لايقبل الإجابة العددية ولكنه يعامل هذه الإجابة كقيمة حرفية حتى لو كانت عددا.

٥ - ٢ - ٢ الأمـر (INPUT)

هذا الأمر يشبه الأمر (ACCEPT) حيث يعرض رسالة للمستخدم وينتظر إجابته على هذه الرسالة. ولكن الأمر (INPUT) يتعامل مع إجابة المستخدم حسب نوعها سواء كانت حرفية أو عددية. فإذا كانت الإجابة عددية يقوم بإنشاء متغير عددى. فمثلا يمكن كتابة السطر التالى :

```
INPUT "Enter your age"  TO  Age
```

وهذا يؤدى إلى ظهور الرسالة التالية على الشاشة :

Enter your age

وينتظر البرنامج من المستخدم إدخال قيمة معينة. وعلى حسب نوع هذه القيمة يقوم بإنشاء متغير ذاكرة من نفس النوع.

٥ - ٢ - ٣ الأمـر (WAIT)

يستخدم هذا الأمر مثل الأمرين السابقين فى عرض رسالة للمستخدم ولكنه ينتظر من المستخدم الضغط على أى مفتاح حتى يستمر تنفيذ البرنامج. ويستخدم عادة عندما يراد إيقاف تنفيذ البرنامج إيقافا مؤقتا (Pause) حتى يستطيع المستخدم قراءة بيانات معينة على الشاشة. وهذا الأمر يمكن كتابته دون كتابة أى شىء بعده. وهذا يؤدى إلى ظهور الرسالة التالية :

Press any key to continue

وهذه هى الرسالة المبدئية (Default).

كما يمكن كتابة أى رسالة أخرى كالآتى مثلا :

WAIT "Press any key To return to main menu "

كما يمكن استخدامه دون عرض أى رسائل وذلك كالآتى :

WAIT "   "

كما يمكن استخدامه فى إنشاء متغير ذاكرة (Pr) يتم فيه تخزين الحرف الذى يكتبه المستخدم وذلك كالآتى مثلا :

WAIT "Send report to printer ? (Y/N)" TO Pr

وهذا المتغير الذى يتم إنشاؤه يكون متغيرا حرفيا. وإذا أريد استخدام هذا الأمر فى استقبال قيمة عددية من المستخدم يتم كتابة الآتى :

```
WAIT "Enter your choice (1-5)" TO choice
Choice = VAL(Choice)
```

وفى هذه الحالة يتم إنشاء متغير ذاكرة حرفى (Choice) يحتوى على الرقم الذى يدخله المستخدم ثم يتم تحويل هذا المتغير إلى قيمة عددية.

٥ - ٢ - ٤ الأمر (@...SAY...GET)

يستخدم هذا الأمر مثل الأوامر السابقة فى عرض رسالة للمستخدم وانتظار الإجابة على هذه الرسالة ثم تخزين هذه الإجابة فى متغير ذاكرة. ولكن هذا الأمر يمتاز بالقدرة على التحكم فى مكان ظهور الرسالة على الشاشة عن طريق الإحداثيات التى يتم كتابتها بعد الحرف (@).

وهذا الأمر يختلف عن الأوامر السابقة فى أنه يلزم قبل استخدامه إنشاء متغير الذاكرة أولا. كما يستخدم الأمر (READ) بعد ذلك فى تخزين إجابة المستخدم فى هذا المتغير الذى سبق إنشاؤه. فمثلا عندما يراد سؤال المستخدم عن الإختيار المطلوب من الشاشة وتخزين هذا الاختيار فى المتغير (Choice) يتم كتابة السطور التالية :

```
Choice = 0
@ 10,5 SAY "Enter choice" GET Choice
READ
```

وهذه الأوامر تبدأ بإنشاء المتغير العددى (Choice) ثم عرض الرسالة (Enter Choice) فى السطر العاشر والعمود الخامس. ثم ينتظر البرنامج من المستخدم إدخال أى قيمة عددية ويقوم بتخزينها فى المتغير (Choice).

٥ - ٣ الحلقة التكرارية

تعد الحلقة التكرارية أحد الأشكال الشائعة الإستخدام فى البرامج بصفة عامة وفى برامج عائلة (DBase) بصفة خاصة. ويتم تكوينها فى برامج عائلة (DBase) بواسطة الأمر (DO WHILE). وهى تبدأ عادة بهذا الأمر وتنتهى بالأمر (ENDDO). ولتوضيح وظيفة الحلقة التكرارية يتم كتابة برنامج إسمه (Count) مثلا. هذا البرنامج يتكون من السطور التالية :

```
CLEAR
SET TALK OFF
STORE 1 TO X
DO WHILE X <= 20
    ? X
    X = X + 1
ENDDO
```

وعند تنفيذ هذا البرنامج يظهر الآتى على الشاشة :

```
1
2
```

3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20

ويمكن توضيح تنفيذ البرنامج لهذه الحلقة التكرارية كالآتى :

يبدأ البرنامج بالأمر (CLEAR) الذى يؤدى إلى مسح الشاشة. ويلى ذلك الأمر (SET TALK OFF) الذى يمنع رسائل البرنامج من الظهور على الشاشة أثناء تنفيذه. ثم يتم إنشاء المتغير (X) بواسطة الأمر (STORE) وإعطاؤه القيمة (1). ثم تبدأ الحلقة التكرارية بالأمر (DO WHILE X <= 20). وهذا الأمر يعنى البدء فى تنفيذ الحلقة التكرارية التى يستمر تنفيذها طالما كانت قيمة (X) أصغر من أو تساوى (٢٠). والأمر (X ?) يؤدى إلى عرض قيمة (X) على الشاشة. كما يستخدم الأمر (X = X + 1) فى زيادة قيمة المتغير (X) بواحد فى كل مرة يتم فيها تنفيذ الحلقة التكرارية. والأمر (ENDDO) يحدد نهاية الحلقة التكرارية.

والأمر (DO WHILE) يستخدم عادة عندما يراد المرور على سجلات ملف قاعدة البيانات وتنفيذ عمليات معينة عليها. وفى هذه الحالة فإن هذا الأمر يتم كتابته عادة كالآتى :

```
DO WHILE .NOT. EOF()
```

وهذا يعني تنفيذ الحلقة التكرارية طالما لم يتم الوصول إلى نهاية الملف (EOF()). وهي إختصار (END OF FILE).

٥ - ٤ إتخاذ القرار بواسطة الأمر (IF)

يستخدم هذا الأمر عندما يراد اتخاذ قرار بناء على معلومات معينة. ويمكن توضيح هذا الأمر عن طريق كتابة برنامج نسميه مثلا (IFTest) ويتم كتابته كالآتي :

```
MODIFY COMMAND IFTest
```

وفي هذه الحالة تظهر شاشة الكتابة ويتم كتابة السطور التالية :

```
********************************* IFTest.prg
* - - - - - Program to test the (IF) Command
CLEAR
ACCEPT "Turn printer on? (Y/N)" TO Pr
* - - - - - If answer is yes , then
IF UPPER(Pr) = "Y"
        SET PRINT ON
        ? "You chose the printer"
        EJECT
        SET PRINT OFF
* - - - - - If answer is not yes , then
ELSE
        CLEAR
        ? "You chose the screen"
ENDIF
```

وأول سطرين من هذا البرنامج هما سطران مخصصان لملاحظات مخطط البرامج. لذلك فإنهما يبدأن بحرف النجمة (*). وهذه الملاحظات تفيد في تذكير مخطط البرامج بإسم البرنامج ووظيفته. كما أنها تفيد داخل البرنامج في تحديد وظيفة كل مجموعة من الأوامر تؤدي وظيفة محددة. والأمر (CLEAR) يؤدي إلى مسح الشاشة.

والسطر التالى يؤدى إلى عرض السؤال (Turn Printer on? (Y/N)) على الشاشة وانتظار إجابة المستخدم ثم تخزين هذه الإجابة فى المتغير (Pr).

والسطر التالى يقوم باختبار الحرف الذى يكتبه المستخدم فإذا كان هذا الحرف بعد تحويله إلى حرف كبير (Uppercase) مساويا للحرف (Y) يتم تنفيذ مجموعة الأوامر المحصورة بين الأمر (IF) وكلمة (ELSE). وفائدة استخدام الدالة (UPPER) هى الحصول على نفس النتيجة سواء أدخل المستخدم الحرف كبيرا (Uppercase) أو صغيرا (Lowercase). وإذا أدخل المستخدم أى حرف آخر غير (Y) أو (y) يقوم البرنامج بتخطى الأوامر المحصورة بين (IF) و (ELSE) وينفذ الأوامر المحصورة بين (ELSE) و (ENDIF). ويمكن اختبار هذا البرنامج بكتابة هذا الأمر من مشيرة النقطة :

DO IFTest

وفى هذه الحالة يتم مسح الشاشة ثم تظهر الرسالة التالية :

Turn Printer on ? (Y/N)

فإذا تم كتابة (Y) والضغط على مفتاح الإدخال يتم طباعة الرسالة التالية على الطابعة :

You chose the printer.

ثم يتم نقل صفحة على الطابعة (EJECT).

وإذا تم كتابة أى حرف آخر غير (Y) أو (y) يقوم البرنامج بمسح الشاشة وعرض الرسالة التالية :

You chose the screen.

وتستخدم كلمة (ENDIF) فى إنهاء مجموعة الأوامر الخاصة بالأمر (IF). ويجب ملاحظة أن كل أمر (IF) له كلمة (ENDIF) خاصة به. أما كلمة (ELSE) فهى إختيارية.

ويمكن استخدام الدالة (IIF) فى تحقيق نفس العمل الذى يؤديه الأمر (IF) مع توفير فى عدد السطور المطلوب كتابتها. فمثلا يمكن كتابة السطر التالى :

```
?IIF ( X < 10, "Less Than" , "Greater Than" )
```

هذا الأمر يختبر قيمة (X) فإذا كانت أصغر من (10) يتم عرض الرسالة (Less than) على الشاشة وإذا كانت أكبر من (10) يتم عرض الرسالة (Greater than) على الشاشة.

ولتنفيذ نفس هذه العملية بواسطة الأمر (IF) يتم كتابة السطور التالية :

```
IF X < 10
        ? "Less Than"
ELSE
        ? "Greater Than"
ENDIF
```

ومن ذلك يلاحظ أن الدالة (IIF) قد وفرت فى عدد السطور.

٥ - ٥ إتخاذ القرار بواسطة الأمر (DO CASE)

الأمر (DO CASE) يعتبر صورة أكثر شمولا من الأمر (IF) حيث أنه يتيح للبرنامج الإختيار بين عدة حالات تبعا لقيمة متغير معين. ولتوضيح ذلك يمكن كتابة برنامج نسميه مثلا (CaseTest). ولتنفيذ ذلك يتم كتابة الأمر التالى من مشيرة النقطة (Dot Prompt).

```
MODIFY COMMAND CaseTest
```

ثم يتم كتابة الأوامر الموضحة بالشكل (٥ - ٢). وعند الإنتهاء يتم تخزين هذا البرنامج بكتابة (^End).

```
****************** CaseTest.prg
*----- Test the DO CASE command
CLEAR
INPUT "Enter a number from 1 to 4" TO X
DO CASE
      CASE X = 1
        ? "You entered one"
      CASE X = 2
        ? "You entered two"
      CASE X = 3
        ? "You entered three"
      CASE X = 4
        ? "You entered four"
      OTHERWISE
        ? "Invalid number"
ENDCASE
```

شكل (٥ - ٢)

والسطر الأول من البرنامج بعد سطرى الملاحظات يؤدى إلى مسح الشاشة. والسطر التالى يؤدى إلى عرض الرسالة التالية على الشاشة :

Enter a number from 1 to 4

ثم ينتظر البرنامج حتى يتم إدخال رقم والضغط على مفتاح الإدخال. فى هذه الحالة يتم تخزين هذا الرقم فى المتغير العددى (X).

ثم يبدأ اتخاذ القرار بناء على قيمة (X). فإذا كانت (X) تساوى (1) يعرض البرنامج الرسالة التالية :

You entered one

ثم يترك باقى الاختيارات.

وإذا كانت (X) تساوى (2) يعرض البرنامج الرسالة التالية :

You entered two

ثم يترك باقى الاختيارات. وهكذا.

أما إذا تم إدخال رقم لايحقق أيا من هذه الاختيارات يتم تنفيذ الأمر التالى لكلمة (OTHERWISE). وهو يؤدى إلى ظهور الرسالة التالية :

Invalid number

٥ - ٦ الكتابة التركيبية للبرامج

من المهم جدا كتابة البرامج بالطريقة التركيبية (Structured) حيث أن هذه الطريقة تسمح لمخطط البرامج بالرجوع إلى البرنامج واختباره وتصحيحه بسهولة كما تسمح لأى مخطط برامج بقراءة البرنامج الذى تمت كتابته بواسطة شخص آخر والقيام بتعديله أو تطويره حسب الحاجة.

ولكتابة البرنامج بهذه الطريقة يجب تنفيذ الآتى :

١ - إستخدام الملاحظات الواضحة التى تشرح وظيفة كل مجموعة من الأوامر التى تؤدى وظيفة محددة.
٢ - تحريك بدايات السطور (Indent) فى الحلقات التكرارية والسطور الخاصة بالأمر (IF) والأمر (DO CASE) حتى تظهر بداية ونهاية السطور بوضوح.
٣ - اختيار الأوامر والمتغيرات التى توضح الوظيفة التى يتم تنفيذها.

ويمكن توضيح فائدة هذه الطريقة عن طريق مقارنة برنامجين أحدهما مكتوب دون مراعاة الطريقة التركيبية (Structured). أنظر الشكل (٥ - ٣). والآخر يتم فيه مراعاة هذه الطريقة. انظر الشكل (٥ - ٤).

وفى البرنامج الأول يلاحظ أن قواعد الكتابة التركيبية غير مطبقة. حيث يلاحظ عدم وجود ملاحظات كافية توضح وظيفة كل مجموعة من الأوامر. كما يلاحظ عدم تحريك السطور للداخل (Indentation) فى السطور التى تحتاج إلى ذلك. وهذا يؤدى إلى عدم وضوح بداية الحلقة التكرارية ونهايتها وكذلك بالنسبة للسطور التالية للأمر (IF) أو الأمر (DO CASE). وعندما تزيد الحلقات التكرارية كما يحدث فى معظم البرامج تصبح عملية اختبار البرنامج وتصحيحه عملية معقدة جدا.

```
* * * * * * * * * * * * * * * * * * Library.prg
* - - - -  Example of an unstructured program
USE Library
DO WHILE .T.
CLEAR
@ 1,20 SAY "Library Management system"
@ 3,25 SAY "1. Add new records"
@ 4,25 SAY "2. Print Reports"
@ 5,25 SAY "3. Edit data"
@ 6,25 SAY "4. Exit"
STORE 0 TO Choice
@ 8,20 SAY "Enter choice(1-4) " GET Choice
READ
IF Choice = 1
APPEND
ELSE
IF Choice = 2
REPORT FORM Library
ELSE
IF Choice = 3
EDIT
IF Choice = 4
RETURN
ENDIF
ENDIF
ENDIF
ENDIF
ENDDO
```

شكل (٥ - ٣)

كما يلاحظ استخدام بعض الأوامر التى قد تعطى معنى مخالفا للواقع مثل الأمر (DO WHILE .T.). وهذا يعنى أن الحلقة التكرارية سيتم تنفيذها إلى مالانهاية.

والحقيقة أنها لاتنفذ إلى مالانهاية لأن اختيار المستخدم للرقم (4) يؤدى إلى توقف تنفيذها.

والبرنامج بهذه الصورة سوف يتم تنفيذه كما أنه سوف يؤدى الوظيفة المطلوبة. ولكنه لايعتبر برنامجا موثقا (Documented) يمكن الرجوع إليه وتعديله أو تطويره.

```
* * * * * * * * * * * * * * * * * Library.prg
* - - - - -  Library system main menu.
USE Library
STORE 0 TO CHOICE

DO WHILE Choice # 4
    CLEAR "
    @ 1,20 SAY "Library Management system "
    @ 3,25 SAY "1. Add new records "
    @ 4,25 SAY "2. Print Reports "
    @ 5,25 SAY "3. Edit data "
    @ 6,25 SAY "4. Exit
    @ 8,20 SAY "Enter choice(1-4) " GET Choice
    READ
    * - -  Branch according to user's request.

    DO CASE
        CASE Choice = 1
         APPEND
        CASE Choice = 2
         REPORT FORM Library
        CASE Choice = 3
         EDIT
    ENDCASE
ENDDO(while choice # 4)

* - - - - -  when choice = 4 exit.
RETURN
```

شكل (٥ - ٤)

أما البرنامج الثانى (الشكل (٥ - ٤)) فإنه يؤدى نفس وظيفة البرنامج الأول ولكنه مكتوب بالطريقة التركيبية. حيث يلاحظ أن كل الملاحظات واضحة تماما ومكتوبة بأسلوب واضح كما يلاحظ أن كل السطور داخل الحلقة التكرارية تم تحريكها للداخل قليلا (Indentation) مما يجعل من السهل تحديد بداية الحلقة التكرارية ونهايتها (يمكن تحريك الأصبع بدءا من الأمر (DO WHILE) رأسيا حتى يصل إلى الأمر (ENDDO) الخاص به). كما يمكن تحديد بداية الأمر (DO CASE) ونهاية السطور الخاصة به بنفس الطريقة. كما تم استبدال الأوامر التى كانت تسبب شيئا من الغموض بأوامر أخرى واضحة. حيث تم استخدام الأمر (DO WHILE Choice # 4) بدلا من الأمر(DO WHILE .T.).

كما يلاحظ استخدام الملاحظات بجانب كلمة (ENDDO) لتوضح لمخطط البرامج أى حلقة تكرارية تتبع لها هذه الكلمة. وهذا يفيد بصفة خاصة عندما تزيد الحلقات التكرارية فى البرنامج. حيث أن أى ملاحظات تكتب بعد كلمة (ENDDO) أو كلمة (ENDIF) لايشعر بها البرنامج وتعامل مثل أى ملحوظة مكتوبة بعد الحرف (*).

الفصل السادس

وسائل التصحيح (Debugging)

٦ - ١ مقــدمــة

يجب أن يكون واضحا أنه نادرا ما يوجد البرنامج الذى يكتب أول مرة بدون أخطاء. والأخطاء عموما قد تكون بسيطة مثل الهجاء الخاطىء للأوامر (Misspelling). وهذه الأخطاء يسهل اكتشافها عند تشغيل البرنامج. وقد تكون أخطاء منطقية (Logical) وهى عادة تكون معقدة ويصعب اكتشافها. حيث أنها قد لاتؤدى إلى إيقاف البرنامج ولكنها تعطى نتيجة غير النتيجة المتوقعة منه.

وبرامج عائلة (DBase) توفر كثيرا من أدوات التصحيح (Debugging Tools) التى تسهل على مخطط البرامج إكتشاف الأخطاء وتصحيحها.

فمثلا عندما يجد البرنامج خطأ يؤثر فى تنفيذ البرنامج فإنه يعرض السطر الذى يحتوى على الخطأ والبرنامج الذى يحتوى على الخطأ مع الرسالة التحذيرية التالية :

Cancel , Ignore or Suspend? (C,I or S)

وهذه الاختيارات تعنى الآتى :

١ - الإختيار (Cancel)

هذا الإختيار يؤدى إلى إنهاء تشغيل البرنامج والعودة إلى مشيرة النقطة (Dot Prompt). وفى هذه الحالة تختفى كل متغيرات الذاكرة الخاصة (Private) التى تم إنشاؤها خلال البرنامج.

٢ - الإختيار (Suspend)

هذا الإختيار يؤدى إلى توقف البرنامج مؤقتا مع ظهور الرسالة (Do suspend). وفى هذه الحالة تظل متغيرات الذاكرة الخاصة موجودة فى الذاكرة. كما يمكن استكمال البرنامج فى أى وقت عن طريق كتابة الأمر (RESUME) من مشيرة النقطة.

٣ - الإختيار (Ignore)

يؤدى هذا الاختيار إلى تخطى السطر المحتوى على الخطأ ومحاولة تنفيذ باقى سطور البرنامج إذا لم تكن معتمدة على هذا السطر.

ويمكن إيقاف البرنامج فى أى لحظة أثناء تنفيذه بالضغط على مفتاح الهروب (Esc). وفى هذه الحالة تظهر نفس الاختيارات الثلاثة السابق شرحها.

٦ - ٢ عرض الذاكرة (Memory)

من الوسائل الفعالة فى اكتشاف أخطاء البرنامج عرض الذاكرة (Memory) أثناء تنفيذ البرنامج.

فعند حدوث خطأ معين مثلا يمكن كتابة الأمر التالى :

```
DISPLAY MEMORY
```

فى هذه الحالة يتم عرض حالة الذاكرة المؤقتة (RAM) من حيث متغيرات الذاكرة الموجودة بها وأسمائها ومحتوياتها وأنواعها. ويمكن فى هذه الحالة إكتشاف سبب الخطأ عن طريق مراجعة أسماء متغيرات الذاكرة وأنواعها.

وللحصول على محتويات الذاكرة مطبوعة على الورق يتم كتابة السطر التالى :

```
DISPLAY MEMORY TO PRINT
```

كما يمكن عرض هيكل ملف قاعدة البيانات المفتوح عن طريق كتابة السطر التالى :

```
DISPLAY STRUCTURE
```

فى هذه الحالة يمكن اختبار أسماء الحقول وأنواعها واكتشاف أى خطأ موجود. كما يمكن استخدام الأمر (DISPLAY STATUS) لعرض أسماء ملفات قواعد البيانات المفتوحة وكذلك ملفات الفهرس.

٦ - ٣ عرض التاريخ (History)

المقصود بالتاريخ (History) هو ذاكرة خاصة يتم فيها تخزين آخر عشرين أمرا تم إدخالها من خلال مشيرة النقطة (Dot Prompt). وهذا يؤدى إلى إمكانية الرجوع إلى أى

أمر تم إدخاله من آخر عشرين أمرا. وذلك بالضغط على مفتاح السهم لأعلى (|) مثلا. وهذا يوفر على مخطط البرامج كتابة الأمر عدة مرات. حيث يكفى الضغط على مفتاح السهم لأعلى (|) عدة مرات حتى يظهر الأمر المطلوب ثم الضغط على مفتاح الإدخال.

ولكن هذا لاينطبق على ملفات الأوامر (Command files). حيث أن أوامر الملف لايتم تخزينها فى ذاكرة التاريخ (History) إلا بعد استخدام الأمر (SET DOHISTORY ON) لتجهيز ذاكرة التاريخ لتخزين الأوامر التى يتم تنفيذها تباعا.

فمثلا يمكن كتابة هذا الأمر قبل تنفيذ البرنامج. فإذا توقف البرنامج نتيجة خطأ معين يمكن استخدام الأمر (SUSPEND) فى إنهاء البرنامج مؤقتا. ثم يتم كتابة الأمر (DISPLAY HISTORY) أو الأمر (LIST HISTORY). وفى هذه الحالة يتم عرض آخر عشرين أمرا تم تنفيذها فى البرنامج وقد يؤدى هذا إلى إكتشاف مكان الخطأ.

ويمكن زيادة عدد الأوامر التى يتم إدخالها فى ذاكرة التاريخ إلى أكثر من عشرين أمرا عن طريق كتابة السطر التالى مثلا :

SET HISTORY TO 50

ويجب ملاحظة أن الأمر (SET DOHISTORY ON) يؤدى إلى إبطاء تنفيذ البرنامج بدرجة كبيرة. ولذلك يراعى عند الإنتهاء من عملية التصحيح (Debugging) إعادة الأمر إلى الوضع المبدئى. وذلك بكتابة السطر التالى :

SET DOHISTORY OFF

٦ - ٤ استخدام الأمـــر (SET TALK ON)

عادة يتم كتابة الأمر (SET TALK OFF) فى أى برنامج حتى لاتظهر الرسائل التى توضح كل خطوة يتم تنفيذها وهذه الرسائل تكون مفيدة جدا عند تصحيح البرنامج. ولذلك يفضل أثناء اختبار البرنامج وتصحيحه كتابة الأمر (SET TALK ON) قبل تشغيل البرنامج. كما يتم متابعة الرسائل التى تظهر على الشاشة عند تنفيذ كل خطوة. فعند حدوث خطأ معين فى البرنامج قد تؤدى هذه الرسائل إلى توضيح سبب هذا الخطأ.

٦ - ٥ استخدام الأمــــر (SET ECHO ON)

عند كتابة هذا الأمر قبل تنفيذ البرنامج فإن هذا يؤدى إلى عرض كل سطر على الشاشة قبل تنفيذه. ويجب إعادة الأمر إلى الوضع المبدئى (Default) عند الإنتهاء من عملية التصحيح وذلك بكتابة السطر التالى :

SET ECHO OFF

٦ - ٦ إستخدام الأمــــر (SET STEP ON)

عند استخدام الأمر (SET ECHO ON) فإن الرسائل التى توضح خطوات التنفيذ تظهر على الشاشة أثناء تنفيذ البرنامج كما سبق الإيضاح. ولكن الرسائل فى هذه الحالة تظهر سريعة مع سرعة تنفيذ هذه الخطوات. فإذا أريد إبطاء ظهور هذه الرسائل حتى يستطيع مخطط البرامج متابعتها فيمكنه فى هذه الحالة استخدام الأمر (SET STEP ON). وهذا يؤدى إلى عرض كل أمر عند تنفيذه ثم توقف البرنامج مؤقتا ثم عرض الرسالة التالية :

Press SPACE to stop,S to suspend,or Esc to cancel

ويمكن فى هذه الحالة لمخطط البرامج الضغط على مسطرة المسافات (Space) لتنفيذ الأمر التالى أو كتابة (S) لتعليق تنفيذ البرنامج وذلك عند ملاحظة خطأ معين مثلا أو الضغط على مفتاح الهروب (Esc) للخروج من البرنامج. وهذا يتيح لمخطط البرامج التحكم فى ظهور رسائل تنفيذ الأوامر أثناء تشغيل البرنامج مما يساعده على اكتشاف الأخطاء المنطقية (Logical Errors) فى البرنامج.

٦ - ٧ إستخدام الأمــــر (SET DEBUG ON)

يستخدم هذا الأمر لإرسال خطوات تنفيذ البرنامج إلى الطابعة. وهو يستخدم مع الأمر (SET ECHO ON) والأمر (SET STEP ON) لمتابعة خطوات البرنامج خطوة خطوة حتى يتم اكتشاف مكان الخطأ.

٦ - ٨ أهم أخطاء كتابة البرامج

فى هذا الجزء يتم توضيح أهم رسائل الأخطاء (Error Messages) التى تظهر أثناء تنفيذ البرنامج ووسائل علاج هذه الأخطاء.

٦ - ٨ - ١ الرسالة (Data type mismatch)

تظهر هذه الرسالة عادة عند معاملة نوع معين من البيانات كنوع آخر. مثل معاملة البيانات الحرفية كعددية مثلا أو البيانات التاريخية كبيانات حرفية وهكذا. فمثلا عند كتابة السطر التالى :

```
LIST FOR Date = "01/30/90"
```

تظهر الرسالة (Data type mismatch) وذلك لأن الحقل (Date) حقل تاريخى يحتوى على تاريخ معين. أما الطرف الأيمن ("01/30/90") فهو بيان حرفى لوجود علامات التنصيص (Quotation Marks). ولعلاج ذلك يجب تحويل التاريخ (Date) إلى القيمة الحرفية المناظرة كالآتى :

```
LIST FOR DTOC(Date) = "01/30/90"
```

٦ - ٨ - ٢ الرسالة (Invalid function argument)

تظهر هذه الرسالة عادة عند استخدام نوع معين من البيانات مع دالة تستخدم نوعا آخر. فمثلا عند كتابة السطر التالى :

```
? UPPER(X)
```

وبفرض أن المتغير (X) يحتوى على قيمة عددية (Numeric). فى هذه الحالة تظهر الرسالة :

```
Invalid function argument
```

وذلك لأن الدالة (UPPER) تستخدم فقط لتحويل المدخلات الحرفية إلى حروف

كبيرة (Uppercase).

٦ - ٨ - ٣ الرسالـــة (Unrecognized command verb)

تظهر هذه الرسالة عادة عند كتابة هجاء الأمر خطأ وفى هذه الحالة يتم مراجعة هجاء الأمر كما يتم مراجعة هيئة هذا الأمر (Syntax). ويمكن استخدام شاشات المساعدة (Help) فى تحديد الهيئة (Syntax) الخاصة بكل أمر.

٦ - ٨ - ٤ الرسالـــة (Variable not found)

تظهر هذه الرسالة عادة عند كتابة إسم متغير ذاكرة لم يسبق تعريفه فى البرنامج. وفى هذه الحالة يمكن استخدام الأمر (DISPLAY MEMORY) لعرض محتويات الذاكرة ومعرفة أسماء المتغيرات الموجودة فى الذاكرة. كما يمكن استخدام الأمر (DISPLAY STRUCTURE) لعرض أسماء الحقول فى ملف قاعدة البيانات المفتوح.

٦ - ٨ - ٥ الرسالـــة (Record out of range)

تظهر هذه الرسالة عند محاولة الذهاب إلى سجل (Record) غير موجود فى الملف. فمثلا عند كتابة السطر التالى :

```
GOTO 99
```

بفرض أن الملف يحتوى على (98) سجلا فقط. فى هذه الحالة تظهر الرسالة المذكورة.

وتظهر هذه الرسالة أيضا عند حدوث خطأ فى ملف الفهرس المفتوح نتيجة عدم فتحه عند إجراء بعض التعديلات فى ملف قاعدة البيانات مثلا. وهذا يتم علاجه بإعادة إنشاء الفهرس باستخدام الأمر (REINDEX).

٦ - ٨ - ٦ الرسالـــة (Too Many Files Open)

تظهر هذه الرسالة عادة عند فتح عدد كبير من الملفات فى نفس الوقت. مع

عدم تغيير عدد الملفات فى ملف مواصفات النظام (Config.sys) ليسمح بفتح هذا العدد من الملفات. فى هذه الحالة يتم تعديل ملف المواصفات وتخصيص عدد الملفات (Files) المطلوبة للبرامج. كما يمكن التحكم فى البرنامج لتقليل عدد الملفات المفتوحة فى وقت معين. ويتم ذلك عن طريق إغلاق كل ملف بمجرد إنتهاء التعامل معه.

كما يمكن استخدام ملف الخطوات (Procedure file) فى الإستغناء عن بعض الملفات. حيث يمكن أن يحتوى ملف الخطوات على عدد من البرامج بحد أقصى ٣٢ برنامجا بدلا من كتابة كل منها فى ملف منفصل.

2

الجزء الثانى

نظام معلومات شئون الطلبة

الفصل السابع

تصميـــم النظـــام

٧ - ١ مقدمة

الهدف من هذا البرنامج هو تصميم نظام معلومات للطلبة يمكن السيطرة عليه من خلال القوائم (Menu Driven). وهذا يعنى أن الشخص الذى يقوم بتشغيل البرنامج لايحتاج إلى معرفة أى شىء عن خصائص برنامج (+DBase III) وربما لايحتاج إلى قدر كبير من المعلومات عن الحاسب. ولكن يمكنه عن طريق مجموعة من الإختيارات الواضحة تنفيذ أى عمليات مطلوبة للحصول على أى معلومات أو تعديل البيانات الموجودة أو مسحها أو ... إلخ.

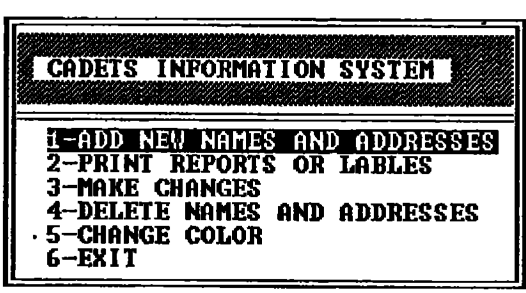
٧ - ٢ تصميم القائمة الرئيسية (Main Menu)

يتكون النظام فى هذه الحالة من أربعة برامج يتم ربطها والسيطرة عليها بواسطة برنامج خامس رئيسى. والبرنامج الرئيسى هو البرنامج الذى يقوم بعرض قائمة الإختيارات للمستخدم. وهذه القائمة تعتبر قلب النظام وهى أول ما يظهر أمام المستخدم وآخر ما يظهر أمامه كما يتم الرجوع اليها دائما بعد تنفيذ كل مهمة.

ويقوم المستخدم بتشغيل النظام بكتابة الأمر

DO Cadets

من مشيرة النقطة. حيث (Cadets) هو إسم البرنامج الرئيسى. وفى هذه الحالة تظهر أمام المستخدم القائمة التالية :



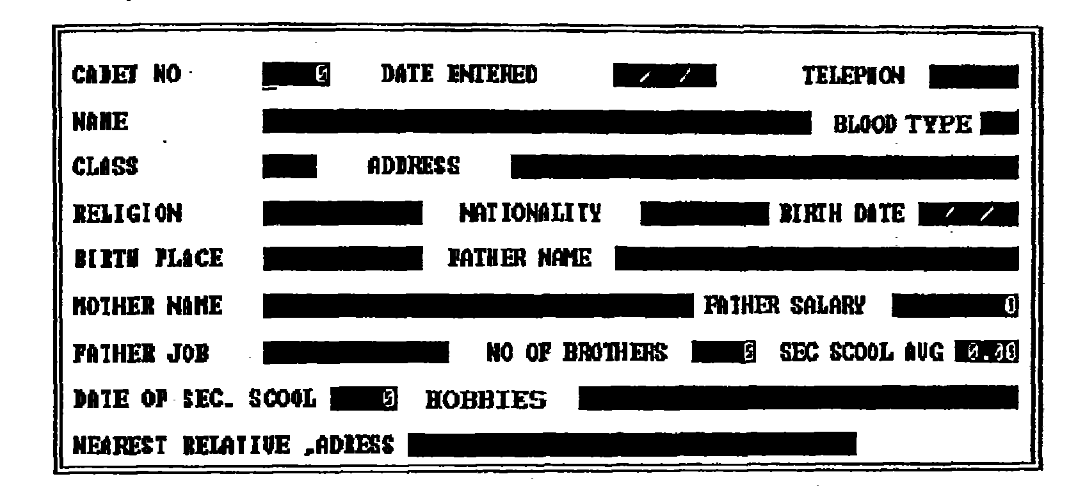
HILIGHT OPTION BY USING ↑ OR ↓ AND PRESS ↵
OR PRESS APPROPRIATE MENU NUMBER

شكل (٧ - ١)

وسوف يتم شرح كل اختيار من هذه الاختيارات والوظائف التى يؤديها قبل الدخول فى تفاصيل البرامج.

٧ - ٢ - ١ إضافة أسماء وعناوين جديدة

وهذا الإختيار يتيح للمستخدم إضافة سجلات جديدة. حيث يقوم المستخدم بكتابة الرقم (1) فتظهر أمامه الشاشة التالية.


CADET NO 0 DATE ENTERED / / TELEPHON
NAME BLOOD TYPE
CLASS ADDRESS
RELIGION NATIONALITY BIRTH DATE / /
BIRTH PLACE FATHER NAME
MOTHER NAME FATHER SALARY 0
FATHER JOB NO OF BROTHERS 0 SEC SCOOL AUG 0.00
DATE OF SEC. SCOOL 0 HOBBIES
NEAREST RELATIVE .ADRESS

شكل (٧ - ٢)

وهى تشمل الحقول التى يتم إدخال البيانات فيها مع وجود مؤشر صغير على أول حقل. وبعد إضافة البيانات تعود القائمة الرئيسية للظهور مرة أخرى.

٧ - ٢ - ٢ طباعة التقارير والعناوين المختصرة

ويتم تنفيذ هذا الإختيار عندما يكتب المستخدم الرقم (٢). وفى هذه الحالة تختفى القائمة الرئيسية وتظهر القائمة الموضحة بالشكل (٧ - ٣).

```
SELECT A REPORT CHOICE

1 - REPORT
2 - LABEIS
3 - RETURN TO MAIN MENU

ENTER YOUR CHOICE<1-3> 3
```

شكل (٧ - ٣)

وعندما يختار المستخدم أحد هذه الإختيارات ، يظهر السؤال التالى :

Do you want (A)ll records, or (Q)uery?

واختيار (All) عن طريق كتابة الحرف (A) يؤدى إلى عرض جميع بيانات الطلبة المخزنة فى قاعدة البيانات. واختيار الإستفهام (Query) عن طريق كتابة الحرف (Q) يؤدى إلى ظهور الرسالة التالية :

Enter name to display

مع ظهور عمود ضوئى (Highlight) يتم خلاله كتابة الحروف الأولى من الإسم المطلوب. ويكفى فى هذه الحالة كتابة أول حرف من الإسم فقط حيث يقوم البرنامج بعرض جميع الأسماء التى تبدأ بهذا الحرف مع أرقام السجلات الخاصة بها. كما يظهر أمام المستخدم السؤال التالى :

Which one do you want?

وهذا يتيح للمستخدم اختيار الإسم المطلوب عن طريق كتابة رقم السجل الخاص به.

وبعد اختيار المستخدم للإسم المطلوب عرض بياناته يظهر السؤال التالى :

Send report to the printer ? (Y/N)

فإذا كتب المستخدم الحرف (Y) يتم طباعة التقرير. وعند كتابة (N) يتم عرضه على الشاشة فقط. ويظهر فى الحالتين التقرير الموضح بالشكل (٧ - ٤).

CADET NO	:	6960
NAME	:	MOHAMED ALY SALEM
CLASS	:	55P
DATE OF ENTERING AIR ACADEMY	:	09/01/85
BLOOD CLASS	:	A
ADDRESS	:	13 - ABBAS ELAKKAD
TELEPHONE NO	:	2603556
RELIGION	:	MUSLIM
NATIONALITY	:	EGYPTIAN
HOBBIES	:	FOOTBALL
BIRTH DATE	:	01/22/67
BIRTH PLACE	:	TANTA
SECONDARY SCHOOL AVERAGE	:	70.00
DATE OF GETTING SECONDARY SCHOOL	:	1985
FATHER NAME	:	ALY SALEM
MOTHER NAME	:	FATMA MAHMOUD
FATHER SALARY	:	600
NOMBER OF BROTHERS	:	5

شكل (٧ - ٤)

٧ - ٢ - ٣ تعديل البيانات (Make Changes)

عندما يختار المستخدم الاختيار رقم (٣) من القائمة الرئيسية تختفى القائمة الرئيسية ويظهر الآتى على الشاشة :

Enter name of person to edit
or just press Return to Quit:

ويستطيع المستخدم إدخال أول حرف فقط من الإسم كما سبق الإيضاح.
وعندما يدخل المستخدم إسما غير موجود فى قاعدة البيانات يظهر الآتى على

الشاشة :

There is no <name>
Press any key to try again

مع ملاحظة أن الإسم الخطأ الذى يدخله المستخدم يظهر مكان (<name>). وعندما يجد البرنامج هذا الإسم تظهر شاشة البيانات الخاصة به ويظهر المؤشر على أول حقل لتعديله حسب الحاجة.

وعندما يكون هناك أكثر من سجل بنفس الإسم أو مبتدئين بنفس الحرف الذى قام المستخدم بإدخاله يقوم البرنامج بعرض الأسماء على المستخدم ومعها أرقام السجلات حتى يقوم باختيار رقم السجل المقابل للإسم المطلوب.

وبعد أن يقوم المستخدم بإجراء التعديل المطلوب على السجل تعود القائمة الرئيسية للظهور من جديد.

٧ - ٢ - ٤ مسح السجلات

عندما يختار المستخدم الاختيار رقم (٤) من القائمة الرئيسية فإن البرنامج يسأل عن الإسم المراد مسح بياناته. ويقوم المستخدم بإدخال الإسم المطلوب أو أول حرف منه فقط كما سبق الإيضاح. وفى هذه الحالة تظهر أمام المستخدم الأسماء التى تشترك فى هذا الحرف ومعها أرقام السجلات حتى يقوم باختيار السجل المطلوب. وفى هذه الحالة يتيح البرنامج للمستخدم التأكد من رغبته فى مسح هذا السجل عن طريق عرض بيانات السجل أمام المستخدم وسؤاله إذا كان يريد مسح هذا السجل أم لا ، وذلك كالآتى :

Record#   name   address

1   Salem Emam   12-dwawin-Cairo

Delete this record?(Y/N)

وعندما يكتب المستخدم الحرف (Y) يظهر السؤال عن الإسم المراد مسح

بياناته مرة أخرى لكى يدخل المستخدم إسما آخر إذا أراد. وعند انتهائه من إدخال كل الأسماء التى يريد مسحها يمكنه الضغط على مفتاح الإدخال بدلا من إدخال إسم جديد. وفى هذه الحالة يتيح له البرنامج التأكد مرة ثانية أنه يريد مسح جميع الأسماء التى أدخلها عن طريق عرض الشاشة التالية :

Records to be deleted..

Record# name address
6 * Eman Salem 12 - Dawawin - Cairo
9 * Medhat Taher 10 - Nasr city - Cairo

Delete all these ? (Y/N)

وعندما يكتب المستخدم (N) فإن البرنامج يتيح له استعادة أى سجل من هذين السجلين حتى لايتم مسحه نهائيا. وتتكرر هذه العملية حتى يتأكد المستخدم تماما من السجلات التى يريد مسحها نهائيا. وفى هذه الحالة يقوم بكتابة (Y) أمام السؤال (Delete all these?) فتختفى هذه السجلات تماما من قاعدة البيانات.

٧ - ٢ - ٥ الخروج من النظام (Exit)

عندما يختار المستخدم الإختيار رقم (٥) فإنه يخرج من البرنامج ويعود إلى مشيرة النقطة (Dot Prompt).

٣ - إنشاء ملف قاعدة البيانات

يتم إنشاء ملف قاعدة البيانات عن طريق قوائم برنامج المساعد (Assistant) أو عن طريق كتابة الأمر (CREATE Cadets) من مشيرة النقطة (Dot Prompt). وذلك كما سبق الإيضاح من خلال الكتابين السابقين. حيث تظهر القائمة الموضحة بالشكل (٧ - ٥) والتى يتم من خلالها تحديد أسماء الحقول (Fields) ونوع كل حقل وعرضه وعدد الأرقام العشرية.

```
CURSOR  <-- -->    INSERT        DELETE        Up a field:    ↑
 Char:    ← →      Char:  Ins    Char:  Del    Down a field:  ↓
 Word: Home End    Field: ^N     Word:  ^Y     Exit/Save:   ^End
 Pan:    ^← ^→     Help:  F1     Field: ^U     Abort:        Esc

      Field Name   Type       Width  Dec    Field Name   Type

   1  NAME         Character    25
   2  JOB          Character    10
   3  ADDRESS      Character    30
   4  AGE          Numeric       2    0
   5  NOTES        Memo         10
   6  PRICE        Numeric       7    2
   7  QTY          Numeric       7    2

MODIFY STRUCTURE |<C:>|MOS1          |Field: 1/7
```

شكل (٧ - ٥)

ونفترض أن هيكل الملف كالآتى :

Field	Field Name	Type	Width	Dec
1	NAME	Character	35	
2	CLASS	Character	4	
3	ADDRESS	Character	40	
4	SEC_SCHOOL	Numeric	5	2
5	DT_ENT	Date	8	
6	T_NO	Character	8	
7	BLOOD	Character	3	
8	RELLIGION	Character	12	
9	NATION	Character	10	
10	B_DATE	Date	8	
11	B_PLACE	Character	12	
12	FATH_NAME	Character	32	
13	MOTH_NAME	Character	33	

Field	Field Name	Type	Width	Dec
14	F_SALARY	Numeric	10	
15	F_JOB	Character	14	
16	NO_BROTHER	Numeric	5	
17	DATE_SEC	Numeric	5	
18	HOBBIES	Character	38	
19	N_RELATIVE	Character	35	
20	NOTES	Memo	10	

ويمكن إنشاء ملف الفهرس أيضا عن طريق برنامج المساعد (Assistant) كما سبق الإيضاح. أو عن طريق كتابة الأمر التالى من مشيرة النقطة :

INDEX ON UPPER(name) TO Name

ويؤدى هذا الأمر إلى إنشاء الملف الفهرسى (Name.ndx).

وتم استخدام الدالة (UPPER) هنا لتحويل الأسماء فى الفهرس إلى حروف كبيرة. ويؤدى هذا إلى توحيد شكل الأسماء داخل الفهرس بصرف النظر عن شكلها فى ملف قاعدة البيانات. وهذا يؤدى إلى سرعة الوصول إلى الإسم المطلوب كما سيتم الإيضاح فيما بعد.

وبعد إنشاء ملف قاعدة البيانات وملف الفهرس الخاص به يتم فتح هذين المللفين أو إغلاقهما من خلال البرنامج كما سيتم الإيضاح. وإذا أراد المستخدم تعديل قاعدة البيانات دون استخدام البرنامج فيجب فى هذه الحالة التأكد من فتح ملف الفهرس مع ملف قاعدة البيانات حتى يتم تحديث الفهرس تبعا لأى تعديل فى قاعدة البيانات. ويتم ذلك عن طريق كتابة الأمر التالى :

USE Cadets INDEX Name

وهذا يؤدى إلى فتح ملف قاعدة البيانات (Cadets.dbf) وملف الفهرس (Name.ndx) فى نفس الوقت.

٧ - ٤ إنشـــاء شـاشـة الإدخـــال

يحتاج البرنامج إلى تصميم شاشة إدخال (Screen) حتى يستطيع المستخدم عن طريقها إدخال البيانات إلى ملف قاعدة البيانات. ويمكن إنشاء شاشة الإدخال عن طريق قوائم المساعد (Assistant) كما سبق الإيضاح. كما يمكن إنشاؤها أيضا بواسطة برنامج يتم التفرع إليه من البرنامج الرئيسى. وكلا الطريقتين سبق شرحهما فى الكتاب رقم (٥) من " مجموعة دلتـــا " ولذلك سنفترض هنا إنشاء شاشة الإدخال الموضحة بالشكل (٧ - ٢) ونفترض أن الملف الخاص بهذه الشاشة إسمه (Cadets.fmt).

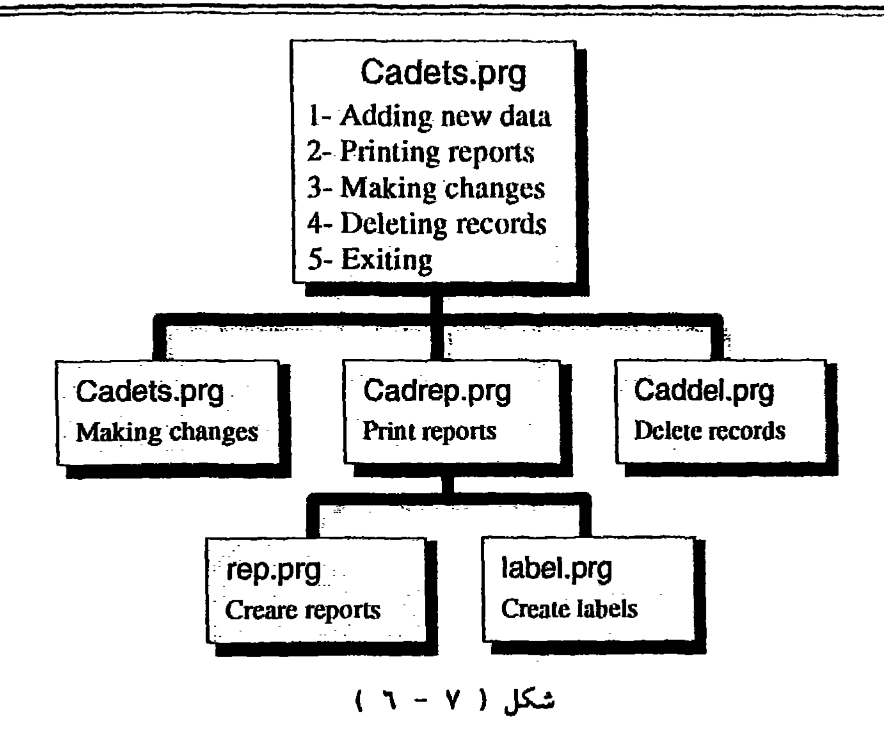
٧ - ٥ إنشـــاء التقـريـر

يمكن إنشاء التقرير عن طريق قوائم برنامج المساعد (Assistant) أيضا أو عن طريق الأمر (CREATE REPORT). وفى الحالتين تظهر القوائم التى سبق شرحها والتى يتم عن طريقها تحديد شكل التقرير المطلوب طباعته. مع ملاحظة أن إنشاء التقرير لايعنى كتابة أى بيانات من ملف قاعدة البيانات فيه ولكنه يحدد فقط الحقول المطلوب إظهارها فى التقرير ومواقع هذه الحقول وأطوالها ثم تظهر البيانات بعد ذلك عند استخدام هذا التقرير فى عرض بيانات سجلات معينة فى الملف.

وهناك طريقة أخرى لتصميم التقرير وذلك عن طريق برنامج فرعى يتم كتابته. وسوف نستخدم هذه الطريقة فى إنشاء التقرير لأنها تحقق مرونة كبيرة فى توزيع الحقول على الشاشة والتحكم فى شكل التقرير. وهذا سيتم شرحه عند دراسة البرامج الفرعية المختلفة.

٧ - ٦ تركيب البرنامج

عند تصميم نظام يحتوى على عدة برامج فمن المهم فى هذه الحالة رسم الشكل الهرمى (Hierarchical Structure) لهذه البرامج. ومع أن ذلك قد يبدو غير مطلوب مع هذا النظام لأنه لايحتوى على العديد من البرامج فإنه من الأفضل دائما التعود على رسم الشكل الهرمى حتى يمكن تتبع كل برنامج على حدة ومعرفة البرنامج الذى يقوم بالإستدعاء (Calling Program) والبرنامج الذى يتم استدعاؤه (Called Program) أنظر الشكل (٧ - ٦).



شكل (٧ - ٦)

ويتضح من هذا الشكل الهرمى أن النظام قد تم تقسيمه إلى عدة وظائف رئيسية وكل وظيفة من هذه الوظائف ينفذها برنامج معين. كما يتضح أيضا أن البرنامج (Cadets.prg) هو البرنامج الذى يقع على قمة الهرم وبالتالى فإنه يسيطر على باقى البرامج.

كما أن كل برنامج فرعى بعد انتهاء مهمته يعود إلى البرنامج الرئيسى ليقوم بتوزيع باقى المهام.

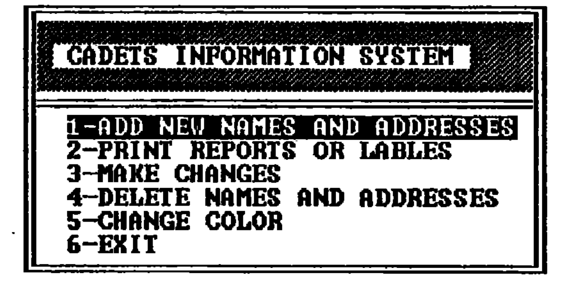
وعملية تقسيم النظام إلى وظائف صغيرة يتم تنفيذها من خلال برامج فرعية تسهل تصميم النظام إلى درجة كبيرة جدا. حيث يصبح من السهل تصميم وتطوير واختبار كل برنامج صغير على حدة. كما أن ذلك يوفر على مخطط البرامج البحث فى عدد كبير من سطور البرنامج عندما يريد تعديل أى جزء من البرنامج أو تصحيحه. حيث يمكنه فى هذه الحالة تحديد المهمة أو الوظيفة التى يريد تعديلها ثم يقوم بتعديل البرنامج المسئول عن تنفيذ هذه المهمة.

الفصل الثامن

البرنامج الرئيسى

يفضل دائما قبل البدء فى تصميم البرنامج كتابة الخطوات المطلوبة باللغة الواضحة بالنسبة لمخطط البرامج سواء كانت اللغة الإنجليزية أو العربية أو أى لغة أخرى وتسمى هذه الخطوات (PSEUDOCODE). وفى البرنامج الرئيسى الخاص بالطلبة (Cadets) تكون هذه الخطوات كالآتى مثلا :

١- تجهيز بيئة البرنامج (Environment).
٢- فتح ملف قاعدة البيانات وملف الفهرس المصاحب له.
٣- إنشاء حلقة تكرارية لعرض القائمة الرئيسية.
٤- مسح الشاشة.
٥- عرض القائمة الرئيسية كالآتى :



HILIGHT OPTION BY USING ↑ OR ↓ AND PRESS↵
OR PRESS APPROPRIATE MENU NUMBER

شكل (٨ - ١)

٦- استقبال اختيار المستخدم.
٧- التفرع إلى الأمر أو البرنامج الذى يحقق اختيار المستخدم.
٨- إعادة عرض القائمة الرئيسية فى حالة عدم اختيار الخروج (Exit).
٩- الرجوع إلى مشيرة النقطة فى حالة اختيار الخروج (Exit).

وبعد الإنتهاء من كتابة الخطوات الأولية (PSEUDOCODE) يتم كتابة البرنامج الذى يحقق هذه الخطوات كالآتى :

```
****************** Cadets.prg
****************** Cadets Information System

* ---------- Set up initial parameters
SET TALK OFF
SET STATUS OFF
SET DEFAULT TO C

* -------- Open the database and index files
USE CADETS INDEX Name

* ---------- Begin loop for main menu
choice = 0
DO WHILE choice < > 5
   CLEAR

         TEXT

         Cadets Information System

       1 - Add new names and addresses
       2 - Print Reports
       3 - Make changes
       4 - Delete names and addresses
       5 - Exit

         ENDTEXT

* ---------- Get users choice
@ 16,20 SAY "Enter choice: " GET choice Picture "9" RANGE 1,5
READ
```

```
*----- Branch to appropriate task orprogram.
DO CASE
   CASE choice = 1
      SET FORMAT TO Cadets
      APPEND
      CLOSE FORMAT
   CASE choice = 2
      DO cadrep
   CASE choice = 3
      DO Cadedit
   CASE choice = 4
      DO Caddel
ENDCASE
ENDDO (while choice < >5)

*------------ Returning to the dot prompt
SET TALK ON
SET STATUS ON
RETURN
```

ويمكن شرح أجزاء البرنامج كالآتى :

يبدأ البرنامج بعرض إسم البرنامج ووظيفته ثم أوامر تجهيز بيئة البرنامج مثل الأمر (SET TALK OFF) الذى يمنع ظهور خطوات التشغيل على الشاشة. ثم الأمر (SET STATUS OFF) الذى يمسح عمود الحالة (Status Bar) من أسفل الشاشة أثناء تنفيذ البرنامج. ثم الأمر (SET DEFAULT TO C) الذى يجعل القرص الصلب هو القرص المستخدم (Default Drive). ويمكن تغيير هذا الأمر إذا كان البرنامج موجودا على قرص مرن باستبدال الحرف (C) بالحرف (A) أو الحرف (B) حسب الحاجة.

والجزء الثانى من البرنامج يفتح ملف قاعدة البيانات (Cadets.dbf) وملف الفهرس المصاحب له (Name.ndx).

والجزء الثالث يقوم بإنشاء الحلقة التكرارية التى يستمر تنفيذها حتى يكتب المستخدم الرقم (5) للخروج من البرنامج. ويتم من خلال هذه الحلقة التكرارية عرض القائمة الرئيسية للبرنامج. والأمر (TEXT) والأمر (ENDTEXT) يعرضان جميع السطور المحصورة بينهما والخاصة بقائمة الاختيارات كما سبق الإيضاح.

ويلاحظ فى بداية الحلقة التكرارية إنشاء متغير الذاكرة (choice) وإعطاؤه القيمة صفر.

والجزء الرابع يقوم باستقبال اختيار المستخدم وتخزينه فى المتغير (choice) الذى سبق إنشاؤه. ويلاحظ استخدام الأوامر (@, SAY, GET, READ). كما يلاحظ استخدام عبارة (Picture "9"). واستخدام الرقم (9) هنا يجبر المستخدم على إدخال أعداد وليس حروف. كما يلاحظ استخدام عبارة (RANGE 1,5). وهذا يجبر المستخدم على إدخال رقم بين (5,1) لأن أى رقم آخر لن يقبله البرنامج.

والجزء الخامس يستخدم الأمر (DO CASE) الذى يساعد البرنامج على إتخاذ القرار بناء على اختيار المستخدم. وعندما يختار المستخدم الرقم (1) فإن البرنامج يفتح شاشة الإدخال الخاصة بملف قاعدة البيانات المفتوح. وذلك من خلال الأمر :

SET FORMAT TO Cadets

كما يسمح للمستخدم بإضافة سجل جديد عن طريق الأمر (APPEND). وبعد الإنتهاء من إدخال هذا السجل يتم إغلاق ملف شاشة الإدخال عن طريق الأمر (CLOSE FORMAT). وفى هذه الحالة تظهر القائمة الرئيسية مرة أخرى.

والاختيارات الأخرى (4 ,3 ,2) تؤدى إلى التفرع إلى برامج فرعية خارجية.

والجزء السادس والأخير من البرنامج يشكل أوامر نهاية الحلقة التكرارية التى يتم تنفيذها فى حالة اختيار المستخدم للاختيار (5). ويلاحظ إضافة تعليق (Comment) بجوار الأمر (ENDDO) يوضح وظيفة الحلقة التكرارية وذلك لأن أى تعليق (Comment) يتم إضافته بعد الأمر (ENDDO) أو الأمر (ENDIF) لايؤثر فى تنفيذ البرنامج. وهذا يكون مفيدا بصفة خاصة عند وجود عدة حلقات تكرارية متداخلة ، حيث يوضح التعليق (Comment) أى (ENDDO) تتبع أى (DO WHILE).

ثم يلاحظ بعد إنتهاء الحلقة التكرارية استخدام الأوامر التى تعيد بيئة الحاسب إلى ما كانت عليه قبل البرنامج. وهى الأوامر التالية :

```
SET TALK ON
SET STATUS ON
RETURN
```

ملاحظة

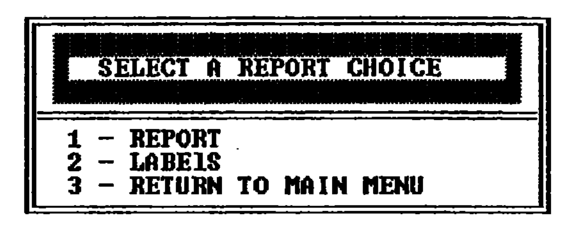
تجدر الإشارة إلى أن المستطيلات التى تظهر فى شاشة القائمة الرئيسية وكذلك العمود المتحرك تتطلب إضافة برنامج فرعى خاص (Routine). ولتبسيط شرح البرنامج للمبتدئين فقد تم حذف هذا البرنامج الفرعى. كما تم شرح هذا البرنامج الفرعى فى نهاية الكتاب للرجوع إليه عندما يراد إضافة أى مؤثرات خاصة فى أى قائمة.

الفصل التاسع

برنامج التقارير

كما سبق الإيضاح فإن برنامج التقارير هو البرنامج (Cadrep.prg) الذى يتم التفرع إليه من البرنامج الرئيسى فى حالة اختيار المستخدم للإختيار رقم (2). وهذا البرنامج يقوم بعرض قائمة اختيارات فرعية تشمل ثلاثة إختيارات لتحديد نوع التقرير المطلوب أو الرجوع إلى القائمة الرئيسية. كما يظهر سؤال عما إذا كان المطلوب عرض التقرير على الشاشة أو طباعته على الطابعة. ويتم توضيح الخطوات الأولية (PSEUDOCODE) كالآتى :

١ - مسح الشاشة وعرض شاشة نوع التقرير المطلوب كالآتى :


SELECT A REPORT CHOICE

1 - REPORT
2 - LABEls
3 - RETURN TO MAIN MENU

ENTER YOUR CHOICE(1-3) 3

شكل (٩ - ١)

٢ - فى حالة عدم اختيار أى نوع يتم الرجوع إلى القائمة الرئيسية مرة ثانية.
٣ - السؤال عما إذا كان المطلوب عرض جميع بيانات قاعدة البيانات فى التقرير أو البحث (Query) عن سجل معين وعرضه.
٤ - عند اختيار البحث (Query) يتم سؤال المستخدم عن الإسم المطلوب البحث عنه.
٥ - يتم البحث عن السجل المطلوب.
٦ - يتم حصر عدد السجلات التى تحتوى على نفس الإسم.
٧ - فى حالة عدم العثور على أى سجل يحتوى على الإسم المطلوب يتم تنبيه المستخدم حتى يدخل إسما آخر.
٨ - فى حالة وجود عدة سجلات تحتوى على نفس الإسم يتم عرض هذه السجلات على المستخدم لاختيار أحدها عن طريق رقم السجل.
٩ - يتم السؤال عما إذا كان المطلوب عرض التقرير على الشاشة أم طباعته على الطابعة.
١٠- يتم طباعة التقرير.
١١- عند اختيار تقرير (Report) يتم التفرع إلى برنامج (Rep.prg).

١٢- عند اختيار تقرير مختصر (Label) يتم التفرع إلى برنامج (Label.prg).
١٣- إيقاف الشاشة مؤقتا (Pause) فى حالة عرض التقرير على الشاشة حتى يستطيع المستخدم قراءة هذا التقرير.
١٤- العودة إلى القائمة الرئيسية مرة ثانية.

بعد الإنتهاء من كتابة الخطوات الأولية (PSEUDOCODE) يتم كتابة البرنامج كالآتى :

```
***********Cadrep.prg
* - -  Reports program for Cadets Information system.

* - - - - - - - Clear screen and ask about report type
CLEAR
TEXT
                              Select a report type
                           1 -  Report
                           2 -  Label
                           3 -  Return to main menu
ENDTEXT

* - - - Initialize variable and ask for report type.
repchoice = 0
@15,20 SAY "Enter your choice (1-3)" GET repchoice ;
  PICTURE "9" RANGE 1,3
READ

* - - - - - - - If return chosen, return to main menu.
IF repchoice = 3
          RETURN
ENDIF

* - - - - - - - Ask about query.
CLEAR
```

```
qchoice = "A"
@ 10,16 SAY "Do you want (A)ll records,or a (Q)uery?";
   GET qchoice PICTURE "!"
READ

* - - - - - - - making Query if requested
IF qchoice = "Q"
        SET EXACT OFF
        qname = SPACE(8)
        @ 16,16 SAY "Enter name to display" GET qname
        READ
        qname = UPPER(TRIM(qname))
        SEEK qname
        recnumb = RECNO()
        COUNT WHILE UPPER(name) = qname TO howmany

        * - - - - -  IF the name is not found, warn the user
        IF howmany = 0
                @ 19,15 SAY "There is no & qname"
                ? CHR (7)
                WAIT "Press any key to return to main menu"
                RETURN
        ENDIF(howmany = 0)

        * - - - - IF there is more than one record having the
        * - - - - same name , they are displayed to the user .
        IF howmany > 1
                CLEAR
                GO TOP
                SEEK qname
                DISPLAY name, class WHILE UPPER(name) = qname
```

```
            @ ROW()+2,15 SAY "which one do you want";
            GET recnumb
            READ
            GOTO recnumb
      ELSE
            SET FILTER TO UPPER(name) = qname
      ENDIF(howmany>1)
ENDIF(qchoice = Q)

* - - - - - - - Ask about printer
toprint = "N"
CLEAR
@ 10,5 SAY "Send records to printer? Y/N" GET toprint  PICTURE "!"
READ

* - - - - - Displaying the report
IF toprint = "Y"
   SET DEVICE TO PRINT
ELSE
   SET DEVICE TO SCREEN
ENDIF
DO CASE
   CASE repchoice = 1
      DO rep
   CASE repchoice = 2
      DO label
ENDCASE

* - - - - - If printer was not selected ,pause
* - - - - - before returning to menu
IF toprint <> "Y"
         @ 24,1 CLEAR
         WAIT "press any key to return to menu ..."
```

```
ENDIF
* - - - - - When report is done, close filter,
* - - - - - and return to main menu
SET FILTER TO
RETURN
```

والبرنامج يبدأ كالعادة بكتابة إسم البرنامج. ثم يبدأ الجزء الأول بمسح الشاشة وعرض قائمة اختيارات نوع التقرير المطلوب. ثم يقوم الجزء الثانى بإنشاء متغير الذاكرة (repchoice) وتخزين القيمة صفر فيه وهذا يجعل المتغير عدديا (numeric). ثم يتم عرض رسالة للمستخدم (prompt) لإدخال الإختيار المطلوب وتخزينه فى المتغير (repchoice) ويتم استخدام الصورة ("PICTURE"9) لإجبار المستخدم على إدخال أعداد. كما يستخدم المدى (RANGE 1,3) لتحديد مدى الأرقام الذى يمكن إدخاله بحيث لايزيد عن (٣) وهذا الجزء يحتوى على السطور التالية :

```
* - - - - - Initialize variable and ask for report type
repchoice = 0
15,20 SAY "Enter your choice (1-3)" ;
      GET rephoice PICTURE "9" RANGE 1,3
READ
```

ويجب ملاحظة أنه عندما يزيد طول الأمر عن عرض الشاشة يتم كتابة الحرف (;) فى نهاية السطر وإستكمال كتابة الأمر فى السطر التالى.

والجزء الثالث من البرنامج يؤدى إلى العودة إلى القائمة الرئيسية فى حالة كتا المستخدم الرقم (3). وهو يحتوى على السطور التالية :

```
IF repchoice = 3
   RETURN
ENDIF
```

والجزء الرابع يسأل المستخدم إذا كان يريد عرض جميع سجلات الملف فى التقرير أو يريد البحث عن سجل معين وعرض بياناته. ويتم تخزين اختيار المستخدم فى المتغير

(qchoice) وهذا الجزء يتكون من السطور التالية :

```
qchoice = "A"
@ 10,16 SAY "Do you want (A)ll records,or a(Q)uery?" ;
        GET qchoice PICTURE "!"
READ
```

والجزء الخامس يقوم بالبحث عن الإسم الذى يدخله المستخدم فى حالة اختيار الحرف (Q). وفى هذه الحالة يتم عرض رسالة للمستخدم لإدخال الإسم المطلوب البحث عنه ثم تخزين هذا الإسم فى المتغير (qname). ويتم تحويل هذا الإسم إلى حروف كبيرة عن طريق الدالة (UPPER). كما يتم مسح المسافات الزائدة عن طريق الدالة (TRIM). ويمكن للمستخدم إدخال حرف واحد أو أكثر من الإسم.

ويتم البحث بواسطة الأمر (SEEK) ثم تخزين رقم السجل الذى يتم الوصول إليه فى المتغير (recnumb) ثم يتم حصر عدد السجلات التى تشترك فى هذا الإسم أو الحرف الذى يتم إدخاله وتخزين هذا العدد فى المتغير (howmany).

ويستخدم الأمر (SET EXACT OFF) حتى تكون المقارنة بين الإسم أو الحرف الذى يتم إدخاله وبين حقل الإسم (name) غير كاملة أى يكفى وجود الحروف فى أول الإسم لتحقيق شرط البحث. ومع أن الوضع المبدئى (Default) هو (SET EXACT OFF) إلا أنه يفضل كتابته فى البرنامج خشية أن يكون قد تم تغيير وضعه المبدئى قبل تنفيذ البرنامج.

ويتكون هذا الجزء من السطور التالية :

```
IF qchoice = "Q"
   SET EXACT OFF
   qname = SPACE(8)
   @ 16,16 SAY "Enter name to display" GET qname
   READ
   qname = UPPER(TRIM(qname))
   SEEK qname
   recnumb = RECNO()
```

```
COUNT WHILE UPPER(name) = qname TO howmany
```

والجزء السادس من البرنامج يقوم بتحذير المستخدم فى حالة عدم العثور على الإسم المطلوب. ويلاحظ هنا استخدام دالة الماكرو فى عرض الإسم أو الحرف الذى أدخله المستخدم داخل السلسلة الحرفية. كما يلاحظ استخدام الدالة CHR(7) لتشغيل جرس التنبيه. ثم تظهر رسالة للمستخدم ليضغط على أى مفتاح للرجوع إلى القائمة الرئيسية مرة أخرى. ويحتوى هذا الجزء على السطور التالية :

```
IF howmany = 0
   @ 19,15 SAY "There is no & qname"
   ? CHR (7)
   WAIT "Press any key to return to main menu"
   RETURN
ENDIF(howmany = 0)
```

والجزء السابع يقوم بعرض بيانات الإسم والفصل الدراسى الخاص بالأسماء التى تشترك فى الإسم أو الحرف المطلوب حتى يقوم المستخدم باختيار رقم السجل الخاص بالإسم المطلوب من هذه السجلات. ويلاحظ أن أمر البحث قد تم استخدامه مرة ثانية وذلك لأن مؤشر السجلات (Record Pointer) يكون قد تحرك نتيجة استخدام الأمر (COUNT) ويراد إعادته مرة ثانية إلى أول سجل يحقق الشرط. ويتم أولا تحريك المؤشر إلى أول الملف عن طريق الأمر (GO TOP) حتى يبدأ البحث من أول الملف. ثم يتم عرض بيانات السجلات عن طريق الأمر (DISPLAY). ويظهر سؤال للمستخدم عن رقم السجل الذى يراد عرض بياناته فى التقرير ويتم تخزين هذا الرقم فى المتغير (recnumb). ثم يتم الذهاب إلى هذا السجل استعدادا لطباعته.

وفى حالة العثور على سجل واحد يحقق الشرط أى أن المتغير (howmany) تكون قيمته (١) أى فى حالة عدم تحقق الشرط (howmany>1) يتم استخدام مرشح (Filter) للحصول على السجل الذى يحقق الشرط. وهذا الجزء يحتوى على السطور التالية :

```
IF howmany > 1
   CLEAR
```

```
   GO TOP
   SEEK qname
   DISPLAY name, class WHILE UPPER(name) = qname
   @ ROW()+2,15 SAY "which one do you want";
            GET recnumb
   READ
   GOTO recnumb
ELSE
   SET FILTER TO UPPER(name) = qname
ENDIF(howmany>1)
ENDIF(qchoice = Q)
```

والجزء الثامن يسأل المستخدم إذا كان يريد طباعة التقرير أو الإكتفاء بعرضه على الشاشة. ويتم تخزين إجابة المستخدم فى المتغير (toprint). ويحتوى هذا الجزء على السطور التالية :

```
toprint = "N"
CLEAR
@ 10,5 SAY "Send records to printer? Y/N" GET toprint  PICTURE "!"
READ
```

والجزء التاسع يؤدى إلى توجيه التقرير إلى الشاشة أو الطابعة حسب اختيار المستخدم الموجود فى المتغير (toprint). فإذا كان هذا الاختيار (Y) يتم توجيهه إلى الطابعة وإذا كان غير ذلك يتم توجيهه إلى الشاشة. ثم يتم تنفيذ البرنامج (rep.prg) عندما يكون المتغير (repchoice) محتويا على الرقم (1). كما يتم تنفيذ البرنامج (label.prg) عندما يكون المتغير (repchoice) محتويا على الرقم (2). وهذا الجزء يحتوى على السطور التالية :

```
IF toprint = "Y"
   SET DEVICE TO PRINT
ELSE
   SET DEVICE TO SCREEN
```

```
ENDIF
DOCASE
   CASE repchoice = 1
      DO rep
   CASE repchoice = 2
      DO label
ENDCASE
```

والجزء العاشر يتم من خلاله إيقاف التقرير على الشاشة مؤقتا (Pause) حتى يستطيع المستخدم قراءة بيانات التقرير ثم يضغط على أى مفتاح للرجوع إلى قائمة التقارير مرة ثانية. ويحتوى هذا الجزء على السطور التالية :

```
IF toprint < > "Y"
   @ 24,1 CLEAR
   WAIT "Press any key to return to menu"
ENDIF
```

والجزء الحادى عشر يتم من خلاله إغلاق المرشح (Filter) حتى يعود الملف إلى حالته الأولى ثم يتم الرجوع إلى القائمة الرئيسية باستخدام الأمر (RETURN).

٩ - ١ البرنـــامج (rep)

يتم كتابة هذا البرنامج بديلا عن استخدام قوائم برنامج المساعد (Assistant) فى تصميم شكل التقرير. حيث يتم تصميم التقرير باستخدام مجموعة من الأوامر (SAY...@) داخل حلقة تكرارية يتم تكرارها حتى نهاية سجلات الملف (EOF). وذلك لعرض التقارير كلها فى حالة طلب المستخدم ذلك. وهذا البرنامج يكون كالآتى :

```
DO WHILE .NOT. EOF()
   CLEAR
   @  2, 2 SAY "Cadet name :"
   @  2,37 SAY NAME
   @  3, 2 SAY "CLASS :"
```

```
@ 3,38 SAY CLASS
@ 4, 2 SAY "ADDRESS :"
@ 4,38 SAY ADDRESS
@ 5, 2 SAY "Secondary school average :"
@ 5,38 SAY SEC_SCHOOL
@ 6, 2 SAY "Date of entering the institute :"
@ 6,38 SAY DT_ENT
@ 7, 2 SAY "Telephone no :"
@ 7,38 SAY T_NO
@ 8, 2 SAY "Blood Type:"
@ 8,32 SAY BLOOD
@ 9, 2 SAY "Religion   :"
@ 9,38 SAY RELIGION
@ 10, 2 SAY "Nationality :"
@ 10,38 SAY NATION
@ 11, 2 SAY "Birth Date :"
@ 11,38 SAY B_DATE
@ 12, 2 SAY "Birth place :"
@ 12,38 SAY B_PLACE
@ 13, 2 SAY "Father name:"
@ 13,38 SAY FATH_NAME
@ 14, 2 SAY "Mother name:"
@ 14,38 SAY MOTH-NAME
@ 15, 2 SAY "Father salary:"
@ 15,38 SAY F_SALARY
@ 16, 2 SAY "Father Job:"
@ 16,38 SAY F_JOB
@ 17, 2 SAY "Number of brothers :"
@ 17,38 SAY NO_BROTHER
@ 18, 2 SAY "Date of getting Sec-School:"
@ 18,38 SAY DATE_SEC
@ 19, 2 SAY "Hobbies"
```

```
  @ 19,38 SAY HOBBIES
  @ 20, 2 SAY "Nearest relatives"
  @ 20,38 SAY N_RELATIVE
  WAIT "Press any key to continue"
  CLEAR
  @ 1,40 SAY "Notes"
  DISPLAY OFF NOTES

  * - - - Ask the user if he want to stop displaying
  KEYPRESS = 0
  @ 22,12 SAY "Press any key to continue or (Q) to stop"
  DO WHILE KEYPRESS = 0
     KEYPRESS = INKEY()
  ENDDO(KEYPRESS)
  IF KEYPRESS = 113 .OR. KEYPRESS = 81
     CLEAR
     RETURN
  ENDIF
  SKIP
ENDDO(while not EOF)
```

ويلاحظ أن هذا البرنامج يحتوى على سطرين لكل حقل. السطر الأول يتم بواسطته عرض إسم هذا الحقل. لذلك يتم كتابة الإسم كسلسلة حرفية (String). والسطر الثانى يتم بواسطته عرض محتويات هذا الحقل. كما يلاحظ من الإحداثيات (X,Y) التى تلى الحرف (@) أن إسم الحقل ومحتويات الحقل يتم عرضهما على سطر واحد على الشاشة لذلك فعند عرض سجل معين على الشاشة تظهر الشاشة الموضحة بالشكل (٩ - ٢) مثلا.

كما يلاحظ أيضا عرض حقل الملاحظات (Notes) فى شاشة مستقلة. وذلك بمسح الشاشة السابقة عند ضغط المستخدم على أى مفتاح.

والجزء الثانى من البرنامج الذى يبدأ بإنشاء المتغير (KEYPRESS) يقوم بعرض

رسالة للمستخدم لإيقاف عرض السجلات فى حالة عرض السجلات كلها. وذلك عندما يريد أن يوقف عرض هذه السجلات والعودة إلى قائمة التقارير مرة ثانية.

ويلاحظ استخدام الدالة (INKEY()) فى اختبار الشفرة الخاصة بالمفتاح الذى يضغط عليه المستخدم. فإذا كانت هذه الشفرة تساوى (81) أو (113) فإن هذا يعنى أن المستخدم قد ضغط على مفتاح (Q) أو مفتاح (q) على الترتيب. وفى هذه الحالة يتم الخروج من البرنامج والعودة إلى قائمة التقارير. أما إذا ضغط على أى مفتاح آخر فإن هذا يؤدى إلى عرض بيانات السجل التالى :

CADET NO	:	6960
NAME	:	MOHAMED ALY SALEM
CLASS	:	55P
DATE OF ENTERING AIR ACADEMY	:	09/01/85
BLOOD CLASS	:	A
ADDRESS	:	13 - ABBAS ELAKKAD
TELEPHONE NO	:	2603556
RELIGION	:	MUSLIM
NATIONALITY	:	EGYPTIAN
HOBBIES	:	FOOTBALL
BIRTH DATE	:	01/22/67
BIRTH PLACE	:	TANTA
SECONDARY SCHOOL AVERAGE	:	70.00
DATE OF GETTING SECONDARY SCHOOL	:	1985
FATHER NAME	:	ALY SALEM
MOTHER NAME	:	FATMA MAHMOUD
FATHER SALARY	:	600
NOMBER OF BROTHERS	:	5

شكل (٩ - ٢)

٩ - ٢ البرنـــامج (label)

يتم كتابة هذا البرنامج لتصميم التقارير المختصرة بدلا من استخدام برنامج المساعد (Assistant). وهو يماثل البرنامج (rep) السابق شرحه ولذلك سوف نكتفى بكتابة سطور البرنامج دون شرحها وهى كالآتى :

```
* - - - - label.prg
```

```
* - - - - Program for making labels
R = 2
GO TOP
DO WHILE .NOT. EOF()
   @ R,2  SAY CADET_NO
   @ R+1,2 SAY NAME
   @ R+2,2 SAY SEC_SCHOOL
   @ R+3,2 SAY ADDRESS
   @ 17,2 SAY "Press any key to continue or + ;
           "(Q) to stop"
   KEYPRESS = 0
   DO WHILE KEYPRESS = 0
      KEYPRESS = INKEY()
   ENDDO(KEYPRESS)
   IF KEYPRESS = 113 .OK. KEYPRESS = 81
      RETURN
   ENDIF
   CLEAR
   SKIP
   ?
ENDDO
RETURN
```

الفصل العاشر

برنامج التصحيح

يساعد هذا البرنامج على وصول المستخدم إلى السجل المطلوب وتصحيحه. وكالعادة يتم أولا كتابة الخطوات الأولية (PSEUDOCODE) كالآتى :

١ - يتم تكوين حلقة تكرارية لتعديل السجلات.
٢ - يتم السؤال عن الإسم أو الحرف المطلوب البحث عنه.
٣ - فى حالة عدم إدخال أى إسم يتم الرجوع إلى القائمة الرئيسية.
٤ - يتم تحويل الإسم إلى حروف كبيرة حتى يماثل ملف الفهرس.
٥ - يتم البحث عن الإسم المطلوب.
٦ - يتم حساب عدد السجلات التى تحتوى على نفس الإسم.
٧ - فى حالة عدم العثور على أى سجل يحتوى على هذا الإسم يتم تحذير المستخدم حتى يدخل إسما آخر.
٨ - فى حالة وجود عدة سجلات تحتوى على نفس الإسم يتم عرض هذه السجلات على المستخدم لاختيار أحدها عن طريق رقم السجل.
٩ - عند الوصول إلى السجل المطلوب يتم عرض بياناته على شاشة الإدخال التى سبق تصميمها.
١٠- يتم العودة إلى القائمة الرئيسية بعد تعديل الحقول المطلوبة.

ثم يتم كتابة البرنامج كالآتى :

```
******* Cadedit.prg
* Lookup and edit data in the cadets database

* ------ Set up a loop for editing records
entering = .T.
DO WHILE entering
   SET EXACT OFF
   GO TOP
   * ------ Ask for name of person to lookup
   CLEAR
   lookup = SPACE(4)
   @ 10,12 SAY " Enter name of person to edit"
   @ 12,12 SAY "or just press Return to exit" ;
         GET lookup
```

```
READ

* - - - - -  If no name entered , skip all
* - - - - -  commands between here and enddo
IF lookup = "  "
   entering = .F.
   LOOP
ENDIF

* - - - - - convert lookup to uppercase to
* - - - - - match index file, and trim it
lookup = UPPER (TRIM(lookup))
SEEK lookup
mrecord = RECNO()
* - - - - -  count how many there are
COUNT WHILE UPPER (NAME) = lookup TO howmany
IF howmany = 0
   @ 20,10 SAY "There is no & lookup"
   @ 22,10 SAY "press any key to try again"
   ? CHR(7)
   WAIT
   mrecord = 0
ENDIF(howmany = 0)
* - - - - -  if more than one record has that
* - - - - -  name, get more information
IF howmany > 1
   CLEAR
   mrecord = 0
   SEEK lookup
   DISPLAY NAME, CLASS WHILE UPPER(NAME) = lookup
   @ ROW()+3,10 SAY "Edit which record # ?" GET ;
```

```
    mrecord PICTURE "9999"
    READ
  ENDIF
  * - - - - - - If there is a record number greater
  * - - - - - - than zero, edit the record
  IF mrecord > 0
    CLEAR
    GOTO mrecord
    SET FORMAT TO cadets
    READ
    CLOSE FORMAT
  ENDIF

ENDDO (WHILE entiring)
RETURN
```

ويبدأ هذا البرنامج بإسم البرنامج ووظيفته كالمعتاد. ثم يتم تكوين حلقة تكرارية تعتمد على المتغير المنطقى (entering). وحيث أن قيمته فى البداية تكون (.T.) أى صحيح لذلك يتم تنفيذ الحلقة التكرارية أول مرة. ثم يتم التحكم من داخل الحلقة فى قيمة المتغير المنطقى (entering) لاستمرار تنفيذ الحلقة أو إيقاف تنفيذها.

وهذا الجزء يتكون من السطور التالية :

```
entering = .T.
DO WHILE entering
```

وفى الجزء الثانى من البرنامج يبدأ تنفيذ الحلقة التكرارية. ويتم إنشاء المتغير (lookup) الذى يكون طوله (4) حروف. وذلك حتى يدخل المستخدم فيه الحروف الأولى من الإسم المطلوب البحث عنه. ثم يتم عرض رسالة للمستخدم لإدخال الحروف المطلوبة وتخزين هذه الحروف فى المتغير (lookup). ويحتوى هذا الجزء على السطور التالية :

```
CLEAR
lookup = SPACE(4)
@ 10,12 SAY " Enter name of person to edit"
@ 12,12 SAY "or just press Return to exit" ;
GET lookup
READ
```

وفى الجزء الثالث يتم تخزين القيمة (.F.) فى المتغير (entering) فى حالة ضغط المستخدم على مفتاح الإدخال دون كتابة أى حروف. وفى هذه الحالة يتم الإنتقال إلى أول الحلقة التكرارية بواسطة الأمر (LOOP). وحيث أن المتغير (entering) يكون غير صحيح (False) فلا يتم تنفيذ الحلقة التكرارية وتعود القائمة الرئيسية للظهور. ويحتوى هذا الجزء على السطور التالية :

```
IF lookup = "  "
    entering = .F.
    LOOP
ENDIF
```

وفى الجزء الرابع من البرنامج يتم البحث عن الحروف التى يدخلها المستخدم خلال حقل الإسم (name). مع ملاحظة أنه سبق فتح ملف الفهرس من خلال البرنامج الرئيسى. كما أن هذا الفهرس قد تم إنشاؤه بحيث يتضمن الأسماء بحروف كبيرة (Upercase) حتى يسهل مطابقته على الإسم الذى يدخله المستخدم. كما يتم تحويل الحروف التى يدخلها المستخدم أيضا إلى حروف كبيرة. لذلك فإن إدخال المستخدم للحروف الكبيرة أو الصغيرة لايؤثر فى البحث عن هذه الحروف. كما أن استخدام الدالة (TRIM) يؤدى إلى إلغاء أى مسافات موجودة بعد الحروف التى يدخلها المستخدم. وهذا يتيح له إدخال حرف واحد مثلا دون الخوف من تأثير المسافات الثلاثة الباقية على البحث.

وهذا الجزء يحتوى على السطور التالية :

```
lookup = UPPER (TRIM(lookup))
SEEK lookup
mrecord = RECNO()
```

ويجب ملاحظة أن الأمر (SEEK) وظيفته توجيه مؤشر السجلات (Record Pointer) إلى السجل المطلوب. كما يتم عن طريق السطر التالى له تخزين رقم هذا السجل فى متغير الذاكرة (mrecord). وذلك لكى يسهل الذهاب إلى هذا السجل بعد ذلك.

وفى الجزء الخامس من البرنامج يتم حساب عدد السجلات التى تحقق شرط الإسم وتخزين هذا العدد فى المتغير (howmany). وحيث أن الملف مفهرس على حقل الإسم (name) لذلك يفضل استخدام (WHILE) فى البحث بدلا من (FOR) حيث أنه يقوم بتجميع السجلات التى تحقق الشرط ولذلك يكون البحث أسرع بواسطة (WHILE).

بعد ذلك يقوم البرنامج باختبار العدد الذى تم تخزينه فى المتغير (howmany). فإذا كان هذا العدد صفرا فإن هذا يعنى أنه لايوجد سجل يبدأ بالحروف التى أدخلها المستخدم. ولذلك تظهر الرسالة المبينة والتى يتم خلالها كتابة الحروف التى أدخلها المستخدم فى المتغير (lookup) باستخدام الماكرو (&) كما يتم تحذير المستخدم عن طريق الدالة (CHR(7)) التى تؤدى إلى تشغيل الجرس. وهذا الجزء يحتوى على السطور التالية :

```
IF howmany = 0
   @ 20,10 SAY "There is no & lookup"
   @ 22,10 SAY "press any key to try again"
   ? CHR(7)
   WAIT
   mrecord = 0
ENDIF(howmany = 0)
```

وفى الجزء السادس يتم دراسة حالة أخرى وهى وجود أكثر من سجل يحقق الشرط أى أن (howmany) أكبر من (١) فى هذه الحالة يقوم البرنامج بعرض بعض بيانات هذه السجلات حتى يستطيع المستخدم تمييز السجل الذى يريد تعديله. كما يتم عرض أرقام السجلات الخاصة بهذه السجلات. ويتم سؤال المستخدم عن السجل المطلوب ليقوم بإدخال رقم هذا السجل. وهذا الجزء يحتوى على السطور التالية :

```
IF howmany > 1
   CLEAR
```

```
  mrecord = 0
  SEEK lookup
  DISPLAY NAME , CLASS WHILE UPPER(NAME) = lookup
  @ ROW()+3,10 SAY "Edit which record # ?" ;
  GET mrecord PICTURE "9999"
  READ
ENDIF
```

وفى الجزء السابع يتم اختبار المتغير (mrecord) فإذا كان أكبر من صفر ، فإن هذا يعنى أنه تم إيجاد أحد السجلات المطابقة سواء من خلال الشرط الأول(howmany = 0) أو من خلال الشرط الثانى (howmany > 1) أو من خلال الحالة الوحيدة المتبقية وهى (howmany = 1) لذلك يتم الذهاب إلى هذا السجل باستخدام الأمر (GOTO). ثم يتم فتح ملف التشكيل (Format file) الذى يؤدى إلى عرض شاشة الإدخال الخاصة بملف قاعدة البيانات المفتوح. وهذا يتيح للمستخدم إدخال التعديلات المطلوبة ثم يتم إغلاق ملف التشكيل مرة ثانية. وهذا الجزء يحتوى على السطور التالية :

```
IF mrecord > 0
  CLEAR
  GOTO mrecord
  SET FORMAT TO cadets
  READ
  CLOSE FORMAT
ENDIF
ENDDO (WHILE entering)
RETURN
```

الفصل الحادى عشر

برنامج مسح السجلات

يساعد هذا البرنامج على وصول المستخدم إلى سجل أو عدة سجلات ومسحها. والخطوات الأولية (PSEUDOCODE) لهذا البرنامج تكون كالآتى :

١ - يتم تكوين حلقة تكرارية لمسح السجلات.
٢ - يتم السؤال عن الإسم المطلوب مسح السجل الخاص به.
٣ - عند عدم إدخال أى إسم يتم الرجوع إلى القائمة الرئيسية.
٤ - يتم تحويل الإسم إلى حروف كبيرة (Uppercase).
٥ - يتم البحث عن الإسم المطلوب.
٦ - يتم حصر عدد السجلات التى تحتوى على نفس الإسم.
٧ - فى حالة عدم العثور على أى سجل يحتوى على هذا الإسم يتم تحذير المستخدم حتى يدخل إسما آخر.
٨ - فى حالة وجود عدة سجلات تحتوى على نفس الإسم يتم عرض هذه السجلات على المستخدم لاختيار أحدها عن طريق رقم السجل.
٩ - عند الوصول إلى السجل المطلوب يتم سؤال المستخدم مرة ثانية للتأكد من رغبته فى مسح هذا السجل.
١٠- عند تأكد المستخدم من رغبته فى مسح السجل يتم وضع علامة على هذا السجل تمهيدا لمسحه.
١١- يتم السماح للمستخدم بإدخال أسماء أخرى حتى ينتهى من تحديد الأسماء التى يريد مسحها.
١٢- يتم حصر عدد السجلات التى تم وضع علامات عليها لمسحها.
١٣- طالما كانت هناك سجلات عليها علامات المسح (Marked for deletion) يتم تنفيذ الآتى :
- يتم عرض بيانات السجلات التى تم وضع علامات عليها.
- يتم التأكد من رغبة المستخدم فى مسح هذه السجلات كلها مسحا دائما (Permanently).
- إذا أراد المستخدم استرجاع بعض هذه السجلات يتم إعطاؤه الفرصة لاستعادة أحد هذه السجلات.
- عندما يتأكد المستخدم من رغبته فى مسح السجلات الباقية يتم مسح هذه السجلات مسحا دائما (Permanently).
١٤ - تعود القائمة الخاصة بالمسح للظهور حتى يمكن تكرار العملية مع سجلات أخرى.
١٥ - بعد الإنتهاء يتم الرجوع إلى القائمة الرئيسية.

بعد كتابة الخطوات الأولية (PSEUDOCODE) يتم كتابة البرنامج الذى يقوم بتنفيذها كالآتى :

```
* - - - - - Set up loop for deleting records .
entering = .T.
DO WHILE entering
   * - - - - - - - Ask for name of person to lookup
   CLEAR
   lookup = SPACE(4)
   @ 10,12 SAY "Enter name of person to delete"
   @ 12,12 SAY "or just press return to exit"
          GET lookup
   READ
   * - - - - - If no name entered , skip all
   * - - - - - commands between here and Enddo
   IF lookup = "  "
     entering = .F.
     LOOP
   ENDIF (lookup = "   ")

   * - - - - - convert lookup to uppercase to match
   * - - - - - index file
   lookup = UPPER(lookup)

   * - - - - Try to find requested name, and
   * - - - - - remember record number
   SEEK lookup
   mrecord = RECNO()

   * - - - - - Count howmany there are
   COUNT WHILE UPPER (NAME) = lookup TO howmany
```

```
* - - - If no record has that name , warn the user
* - - - - -  to try again
IF howmany = 0
   @ 20,10 SAY "There is no & lookup"
   @ 22,10 SAY "press a key to try again"
   ? CHR(7)
   WAIT " "
   mrecord = 0
ENDIF (howmany = 0)

* - - - - -  If more than one record has that name
* - - - - -  display records to the user.
IF homany > 1
   CLEAR
   mrecord = 0
   SEEK lookup
   LIST NAME , CLASS WHILE UPPER(NAME) = lookup
   @ ROW()+3,10 SAY "Delete which one?" ;
      GET mrecord PICTURE "9999"
   READ
ENDIF

* - - - - -If the value of (mrecord) greater than
* - - - - - zero double check then delete
IF mrecord > 0
   GOTO mrecord
   CLEAR
   DISPLAY NAME , CLASS
   ?
   WAIT "Delete this record? (Y/N)" TO answer

   * - If answer is yes, mark record for deletion
```

```
      IF UPPER(answer) = "Y"
         DELETE RECORD mrecord
      ENDIF(answer)
   ENDIF(mrecord > 0)
ENDDO (while entering)

* - Before exiting , verify deletion and pack
COUNT FOR DELETED() TO nodels
oktopack = "N"
DO WHILE oktopack = "N" .AND. nodels > 0
        CLEAR
        ? "Records to be deleted .."
        ?
        DISPLAY NAME , CLASS FOR DELETED()
        @ 23,1 SAY "Delete all these? (Y/N)" ;
            GET oktopack PICTURE "!"
        READ

        IF oktopack < > "Y"
           * - if not ok to pack , recall a record
           delrec = 0
           @ 23,1 SAY "Recall which one ;
             GET delrec PICTURE "9999
           READ

           * - if record number entered and record
           * - is indeed deleted, recall it
           IF delrec > 0
                GOTO delrec
                IF DELETED()
                        RECALL RECORD delrec
                        nodels = nodels -1
```

```
            ENDIF
          ENDIF
       ELSE

          * - - - -  if ok to pack , pack and show
          * - - - - -  the process of packing
          SET TALK ON
          PACK
          SET TALK OFF
       ENDIF (oktopack)
ENDDO(oktopack)
RETURN
```

الهدف من هذا البرنامج هو البحث عن سجل معين ومسحه. والجزء الخاص بالبحث لايختلف عن برنامج التصحيح أو برنامج التقارير حيث يقوم البرنامج بعرض رسالة للمستخدم لإدخال الإسم المطلوب البحث عنه ثم البحث عن هذا الإسم باستخدام أوامر البحث المعروفة ثم عرض بيانات السجلات التى تشترك فى هذا الإسم حتى يقوم المستخدم باختيار سجل محدد منها عن طريق رقم السجل.

ولكن المطلوب من البرنامج بعد ذلك مسح هذا السجل. وعملية المسح تتسم بشىء من الخطورة حيث أن المستخدم قد يمسح سجلا ثم يكتشف بعد ذلك أنه قد مسح سجلا مطلوبا عن طريق الخطأ. ولذلك فإن البرنامج يجب أن يتيح للمستخدم عدة مراحل من الإختبار والتحذير حتى يتأكد تماما أن هذا السجل هو السجل المطلوب مسحه.

والجزء الأول من البرنامج لايحتاج إلى إعادة شرحه حيث أنه سبق شرحه فى برنامج التصحيح وبرنامج التقارير.

والجزء الجديد هو الذى يبدأ بعد الوصول إلى رقم السجل المطلوب مسحه وهو الجزء الذى يبدأ بمجموعة السطور التالية :

```
* - - - - - If the value of (mrecord) greater than zero,
* - - - - -  double check then delete
```

```
IF mrecord > 0
   GOTO mrecord
   CLEAR
   DISPLAY NAME , CLASS
   ?
   WAIT "Delete this record? (Y/N)" TO answer

   * - - If answer is yes, mark record for deletion
   IF UPPER(answer) = "Y"
      DELETE RECORD mrecord
   ENDIF(answer)
ENDIF(mrecord > 0)
ENDDO (while entering)
```

ويستطيع المستخدم من خلال الحلقة التكرارية الأولى (WHILE entering) تحديد عدة سجلات مطلوب مسحها. وفى كل مرة يتم عرض بيانات السجل عليه حتى يتأكد أن هذا هو السجل المطلوب. وهذه العملية تؤدى فى النهاية إلى وضع علامات أمام عدة سجلات تمهيدا لمسحها.

وبعد ذلك يتم تكوين حلقة تكرارية أخرى تتيح للمستخدم المسح النهائى لهذه السجلات. وهذه الحلقة التكرارية تظهر فى السطور التالية :

```
* - - - - - Before exiting , verify deletion and pack
COUNT FOR DELETED() TO nodels
oktopack = "N"

DO WHILE oktopack = "N" .AND. nodels > 0
   CLEAR
   ? "Records to be deleted .."
   ?
   DISPLAY NAME , CLASS FOR DELETED()
   @ 23,1 SAY "Delete all these? (Y/N)";
```

```
      GET oktopack PICTURE "!"
   READ
   IF oktopack < > "Y"

   * - - - - - if not ok to pack , recall a record
   delrec = 0
   @ 23,1 SAY "Recall which one (by record#)";
     GET delrec PICTURE "9999"
   READ

   * - - - - -if record number entered and record is
   * - - - - -  indeed deleted, recall it
    IF Delrec > 0
      GOTO delrec
      IF DELETED()
          RECALL RECORD delrec
          nodels = nodels -1
      ENDIF
    ENDIF
   ELSE

    * - - - - -  if ok to pack , pack and show
    * - - - - -  the process of packing
    SET TALK ON
    PACK
    SET TALK OFF
   ENDIF (oktopack)
ENDDO(oktopack)
RETURN
```

وهذه المجموعة من السطور تبدأ بحساب عدد السجلات التى تم وضع علامات عليها باستخدام الأمر (COUNT) وتخزين هذا العدد فى المتغير (nodels). ويجب ملاحظة أن

الأمر (COUNT) يقوم بإنشاء المتغير آليا أى ليست هناك حاجة لإنشاء المتغير (nodels) قبل استخدامه. ثم تبدأ الحلقة التكرارية ويلاحظ قبلها إنشاء المتغير (oktopack) وإعطاؤه القيمة ("N") وذلك حتى يبدأ تنفيذ الحلقة التكرارية مرة واحدة بصرف النظر عن القيمة التى يدخلها المستخدم بعد ذلك فى هذا المتغير. ويتوقف تنفيذ الحلقة التكرارية على شرطين. الأول هو (oktopack = "N") وهذا يكون صحيحا فى البداية. والشرط الثانى هو (nodels > 0) وهذا يعنى أن هناك سجلات تم وضع علامات عليها. أما إذا لم تكن هناك سجلات من هذا النوع فهذا معناه أن المستخدم لم يدخل أى إسم فى البداية لمسحه أو أن الإسم الذى تم إدخاله لم يتم العثور عليه. وفى هذه الحالة لايتم تنفيذ الحلقة التكرارية ويعود البرنامج إلى القائمة الرئيسية.

وعندما يتحقق الشرطان تبدأ الحلقة التكرارية فى التنفيذ وتظهر بيانات الإسم والفرقة (CLASS) الخاصة بالسجلات التى تم وضع علامات عليها. كما يظهر سؤال للمستخدم إذا كان يريد مسح هذه السجلات كلها أم يريد استعادة أحدها. فإذا أراد استعادة أحدها فإن الشرط الموجود مع الأمر (IF oktopack < > "Y") يتحقق. وهذا يؤدى إلى ظهور سؤال آخر للمستخدم عن رقم السجل الذى يريد إستعادته مع ملاحظة أن السجلات تكون معروضة على الشاشة ومعها أرقام السجلات. وعندما يدخل المستخدم رقم السجل المطلوب استعادته يتم تخزين هذا الرقم فى المتغير (delrec). ونتيجة لذلك فإن الشرط (IF delrec >0) يتحقق وينتقل المؤشر إلى هذا السجل. ثم يتم اختبار هذا السجل عن طريق الدالة (DELETED()) للتأكد أن السجل تم وضع علامة عليه لمسحه. وفى هذه الحالة يتم استعادة هذا السجل باستخدام الأمر (RECALL) ثم يتم إنقاص عدد السجلات الموجودة فى المتغير (nodels) بمقدار (١) وتتكرر هذه العملية حتى يقوم المستخدم باستعادة جميع السجلات التى لا يريد مسحها. فإذا كان هناك سجلات باقية يراد مسحها فإن المستخدم يجيب على السؤال الخاص بمسح جميع السجلات بكتابة (Y) وهذا يؤدى إلى عدم تحقق الشرط (IF oktopack < > "Y") وبالتالى تنفيذ الأوامر بعد (ELSE). ومن خلال هذه الأوامر يتم إتمام عملية المسح باستخدام الأمر (PACK). ويلاحظ أنه تم استخدام الأمر (SET TALK ON) قبل استخدام الأمر (PACK) وذلك حتى يرى المستخدم الرسالة الدالة على إتمام عملية المسح. ثم يتم الرجوع إلى البرنامج الرئيسى باستخدام الأمر (RETURN).

3

الجزء الثالث

نظـــام المخـــازن

الفصل الثاني عشر

توصيف النظام

من النظم الشائعة الإستخدام فى الوقت الحالى نظم التحكم فى المخزون (Stock Control) وهى النظم التى تتيح التحكم فى كمية البضائع المخزونة ومتابعة البضائع الواردة والبضائع المنصرفة. واستخدام قواعد البيانات فى هذه النظم أدى إلى كفاءة ملحوظة فى السيطرة على هذه العملية. وقد روعى فى هذا الكتاب شرح برنامج مخازن متكامل لأنه يوفر لمخطط البرامج شرحا لمهارات متعددة فى كتابة البرامج حيث أنه يستخدم نظام التعديل المجمع (Batch Updating) من خلال ملفات الحركة (Transaction Files) كما يمكن ربط هذا البرنامج ببرنامج حسابات العملاء (Accounts Receivable) المشروح فى الجزء الرابع لإنشاء نظام متكامل.

١٢ - ١ تصميم النظام

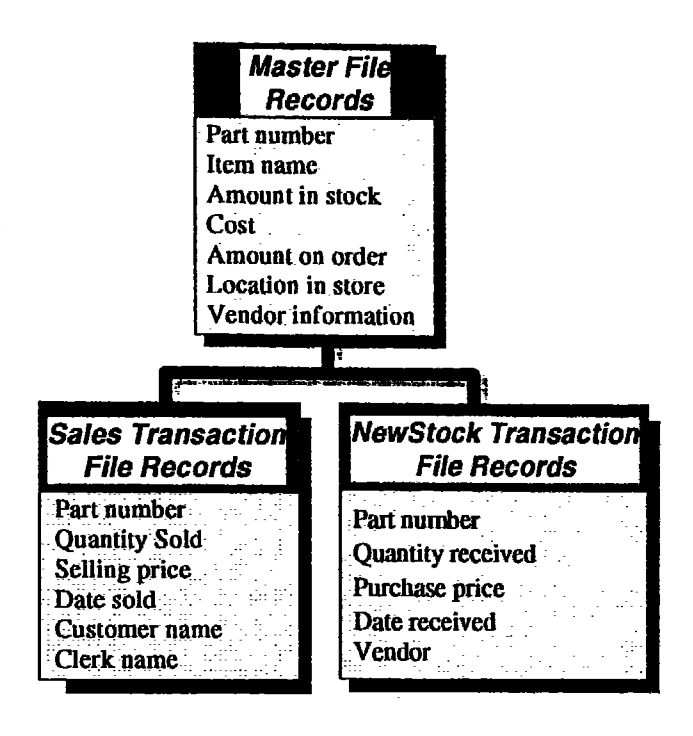
يتطلب نظام المخازن بصفة عامة تصميم عدة ملفات للبيانات أولها يسمى الملف الرئيسى (Master File). وهذا الملف الرئيسى يشتمل على بيانات عن الأصناف الموجودة حاليا بالمخزن مثل كمية هذه الأصناف بالإضافة إلى سعر كل صنف والكمية تحت الطلب من هذا الصنف (In Order) وحد الطلب له (Reorder Point) وإسم البائع (Vendor) الذى يتم شراء هذا الصنف منه وعنوانه. كما يمكن أيضا أن يتضمن بيانات عن مكان كل صنف فى المخزن حتى يسهل على المستخدم الوصول إليه.

ويتطلب نظام المخازن أيضا تصميم ملفات تتضمن بيانات عن حركة الأصناف الموجودة فى المخزن وتسمى ملفات الحركة (Transaction Files). وأحد هذه الملفات يشمل بيانات عن حركة بيع الأصناف مثل بيانات العميل ومتى تم البيع وبأى كمية ورقم فاتورة الشراء ويسمى هذا الملف ملف المبيعات (Sales File).

وهناك ملف حركة آخر يتضمن بيانات عن الأصناف الجديدة التى يتم إدخالها إلى المخزن (Newstock File).

ويمكن ملاحظة العلاقة بين هذه الملفات الثلاثة من الشكل (١٢ - ١)

ويلاحظ من الشكل أن هناك حقلا مشتركا بين الملفات الثلاثة وهو حقل رقم الجزء (Part number) وهذا الحقل يسمى حقل المفتاح (Key Field) الذى يتم عن طريقه ربط الملفات الثلاثة.



شكل (١٢ - ١)

١٢ - ٢ حقل المفتاح (Key Field)

كما سبق الإيضاح فى الجزء الأول من الكتاب فعند تصميم نظام يتعامل مع عدة ملفات قواعد بيانات يجب تحديد حقل مشترك بين هذه الملفات حتى يمكن ربط هذه الملفات من خلاله. وهذا الحقل يجب أن يكون منفردا (Unique) لكل سجل فى الملف الرئيسى (Master File). كما يجب أن يكون بنفس الإسم والنوع والطول فى الملفات الثلاثة.

وفى معظم نظم المعلومات يكون هناك رقم أو كود يمكن استخدامه كحقل مفتاح لأنه يكون منفردا (Unique) . ففى البنوك مثلا يستخدم رقم الحساب كحقل مفتاح حيث لايوجد عميلان يشتركان فى رقم حساب واحد. وكذلك على مستوى الدولة يستخدم الرقم الشخصى (Social Security Number) كمفتاح للحصول على بيانات أى شخص. وبالنسبة لنظام المخازن فإن أنسب رقم يمكن استخدامه كحقل مفتاح هو رقم الجزء حيث أن كل جزء له رقم جزء (Part Number) منفرد (Unique).

١٢ - ٣ وظائف النظام

يجب تصميم نظام يسمح للمستخدم بمتابعة البضائع المخزونة والبضائع تحت الطلب ومكان هذه البضائع فى المخزن. وعند إدخال أى أصناف غير موجودة أصلا بالمخزن يتم تحديد رقم الجزء (Part Number) الخاص بها وإضافة كمية كل صنف إلى الملف الرئيسى. كمايجب أن يوفر النظام التقارير التى توضح حالة الأصناف بالمخزن وتحدد الأصناف التى يجب طلبها كما يقوم بإنشاء طلبات الشراء آليا.

كما يجب أن يسمح النظام للمستخدم بمتابعة حركة الأصناف من حيث الأصناف التى يتم بيعها والشخص الذى يقوم بعملية البيع والعميل الذى تباع له الأصناف وتاريخ البيع ورقم الفاتورة. كما يجب أن يسمح أيضا بمتابعة الأصناف الواردة لتعويض الأصناف الناقصة فى المخزن. كما يجب أن يقوم النظام آليا بتحديث البيانات الموجودة فى الملف الرئيسى (Master File) من ملف المبيعات (Sales) وملف الوارد (New Stock).

١٢ - ٤ تحديد حقول الملفات

فى معظم الأحيان تكون أسهل وسيلة لتحديد حقول ملف قاعدة البيانات هى تحديد المخرجات المطلوبة من هذا الملف وهى التقارير المطلوب إنشاؤها. فمثلا لتحديد حقول الملف الرئيسى (Master File) يتم دراسة محتويات التقارير المطلوبة من هذا الملف. فأحد هذه التقارير هو التقرير الذى يوضح موقف أو حالة الأصناف الموجودة فى المخزن. هذا التقرير يشتمل على رقم الجزء وإسم الجزء والكمية الموجودة من هذا الجزء فى المخزن وسعر الشراء ومكان هذا الجزء فى المخزن والكمية تحت الطلب من هذا الجزء.

وهناك تقرير آخر يسمى تقرير إعادة الطلب (Reorder Report) يتم من خلاله عرض قائمة بالأصناف المطلوبة. كما أن هناك تقريرا آخر يسمى تقرير تحت الطلب (On Order Report) ويعرض قائمة بالأصناف الجارى طلبها. كما أن هناك طلبات الشراء التى يقوم المستخدم بإصدارها لشراء أصناف جديدة.

ولتوفير هذه المخرجات يجب أن يشتمل الملف الرئيسى (Master File) على البيانات التالية :

Part number
Item name
Quantity in stock
Purchase price
Reorder point
Quantity on order
Location in warehouse
Vendor name
Vendor address
Date of last update
Date of last order
Quantity to order

كما يجب أن يوفر النظام أيضا تقارير توضح كل حركة بيع. وهذه التقارير تحتوى على رقم الجزء والكمية ورقم الفاتورة (Invoice Number) وإسم الشخص القائم بالبيع وإسم العميل وسعر البيع وتاريخ البيع. ولذلك فإن ملف المبيعات (Sales File) يجب أن يحتوى على البيانات الآتية :

Part number
Invoice number
Salesperson's name
Quantity sold
Selling price
Date sold
Whether posted or not

كما أن النظام يجب أن يوفر تقارير توضح حركة الأصناف الواردة وتعرض رقم الجزء والكمية الواردة وثمن الشراء وتاريخ الورود وإسم البائع (Vendor). لذلك فإن ملف الأصناف الواردة (New Stock File) يجب أن يحتوى على البيانات الآتية :

Part number
Quantity

Purchase price
Date received
Vendor name
Whether posted or not

١٢ - ٥ تصميم قاعدة البيانات

كما سبق الإيضاح فإن نظام المخازن فى العادة يحتوى على ثلاثة ملفات قواعد بيانات. وهى الملف الرئيسى وليكن إسمه (Master.dbf) وملفين حركة أحدهما خاص بالمبيعات ويسمى (Sales.dbf) والآخر خاص بالأصناف الواردة ويسمى (Newstock.dbf).

١٢ - ٥ - ١ إنشاء الملف الرئيسى (Master File)

يتم إنشاء الملف الرئيسى عن طريق قوائم برنامج المساعد (Assistant) كما سبق الإيضاح فى الكتاب رقم (٥) من مجموعة كتب " دلتا ". كما يمكن إنشاؤه أيضا باستخدام الأمر (CREATE) وذلك كالآتى :

CREATE MASTER

وفى هذه الحالة تظهر شاشة هيكل ملف قاعدة البيانات (Structure) التى يتم عن طريقها كتابة إسم كل حقل ونوعه وطوله وعدد الأرقام العشرية إن وجدت. ويتم تكوين الملف كالآتى :

Field	Field Name	TYPE	WIDTH	DEC
1	PART_NO	Character	5	
2	P_NAME	Character	20	
3	QTY	Numeric	7	2
4	COST	Numeric	9	2
5	REORDER	Numeric	7	2
6	ON_ORDER	Numeric	7	2
7	LOCATION	Character	5	
8	VENDOR	Character	25	

Field	Field Name	TYPE	WIDTH	DEC
9	VENDOR_ADD	Character	25	
10	DATE	Date	8	
11	ORDER_DATE	Date	8	
12	NEW_ORDER	Numeric	7	2

ومن هذا الشكل يلاحظ أن الحقل رقم (١) هو الحقل الخاص برقم الجزء (Part Number) وهو حقل حرفى يتكون من خمسة حروف. والحقل رقم (٢) هو حقل إسم الجزء (Part Name) وهو حقل حرفى يتكون من عشرين حرفا. والحقل رقم (٣) هو حقل الكمية (Quantity) وهو حقل عددى يتكون من سبعة أرقام متضمنا رقمين عشريين. والحقل رقم (٤) هو حقل سعر الشراء (Cost) وهو حقل عددى يتكون من تسعة أرقام متضمنا رقمين عشريين. والحقل رقم (٥) هو حقل حد الطلب (Reorder Point) ويمثل الحد الذى يجب عنده إعادة طلب هذا الصنف ويتكون من سبعة أرقام متضمنا رقمين عشريين. والحقل رقم (٦) هو حقل الكمية الجارى طلبها وهو حقل عددى مكون من سبعة أرقام متضمنا رقمين عشريين. والحقل رقم (٧) هو حقل مكان الصنف فى المخزن (Location) وهو حقل حرفى مكون من خمسة حروف. والحقل رقم (٨) هو حقل البائع (Vendor) وهو حقل حرفى مكون من خمسة وعشرين حرفا. والحقل رقم (٩) هو حقل عنوان البائع وهو حقل حرفى مكون من خمسة وعشرين حرفا. والحقل رقم (١٠) هو حقل تاريخ آخر تحديث للملف وهو حقل تاريخى مكون من ثمانية حروف. والحقل رقم (١١) هو حقل تاريخ آخر طلب للصنف (Date of last order). والحقل رقم (١٢) هو حقل الكمية التى يجب طلبها من هذا الصنف وهو حقل عددى مكون من سبعة أرقام متضمنة رقمين عشريين.

ولإنشاء ملف الفهرس الخاص بهذا الملف على أن يكون الحقل الخاص برقم الجزء هو الحقل الفهرسى (Key Field) يتم كتابة السطرين التاليين من مشيرة النقطة.

```
USE Master
INDEX ON Part_no TO Master
```

١٢ - ٥ - ٢ إنشاء ملف المبيعات (Sales File) ،

يتم إنشاء ملف المبيعات (Sales.dbf) بنفس الطريقة كما سبق إنشاء الملف الرئيسى. ويجب ملاحظة أن حقل رقم الجزء (Part_no) يجب أن يكون موجودا فى هذا الملف وبطول خمسة حروف مثل الملف الرئيسى تماما. ويتم تكوين الملف كالآتى :

Field	Field Name	TYPE	WIDTH	DEC
1	PART_NO	Character	5	
2	INVOICE_NO	Numeric	6	0
3	CLERK	Character	12	
4	CUSTOMER	Character	12	
5	QTY	Numeric	7	2
6	PRICE	Numeric	9	2
7	DATE	Date	8	
8	POSTED	Logical	1	

والحقل رقم (١) هو حقل رقم الجزء (Part number). والحقل رقم (٢) هو حقل رقم الفاتورة (Invoice number) وهو حقل عددى مكون من ستة أرقام. والحقل رقم (٣) هو حقل إسم الموظف القائم بالبيع وهو قد يكون إسم الموظف أو كود معين خاص به حسب الحاجة وهو حقل حرفى مكون من (١٢) حرفا. والحقل رقم (٤) هو حقل إسم العميل (Customer) وهو حقل حرفى مكون من (١٢) حرفا. والحقل رقم (٥) هو حقل الكمية المباعة وهو حقل عددى مكون من سبعة أرقام متضمنة رقمين عشريين. والحقل رقم (٦) هو حقل سعر البيع وهو حقل عددى مكون من تسعة أرقام ورقمين عشريين. والحقل رقم (٧) هو حقل تاريخ البيع وهو حقل تاريخى مكون من ثمانية حروف. والحقل رقم (٨) هو حقل الترحيل (Posted) وهو حقل منطقى (Logical) يستخدم للتحكم فى ترحيل بيانات الصنف إلى الملف الرئيسى وضمان عدم تكرار ترحيلها.

ولإنشاء ملف الفهرس الخاص بهذا الملف يتم كتابة السطرين التاليين من مشيرة النقطة (Dot Prompt)

```
USE Sales
INDEX ON Part_no TO Sales
```

١٢ - ٥ - ٣ إنشاء ملف الأصناف الواردة

يتم إنشاء ملف الأصناف الواردة (NewStock.dbf) بنفس الطريقة كما سبق الإيضاح مع ملاحظة ضرورة وجود حقل رقم الجزء (Part Number) بنفس الإسم والطول والنوع. ويتم تكوين الملف كالآتى :

Field	Field Name	TYPE	WIDTH	DEC
1	PART_NO	Character	5	
2	QTY	Numeric	7	2
3	COST	Numeric	9	2
4	DATE	Date	8	
5	VENDOR	Character	50	
6	POSTED	Logical	1	

والحقل رقم (١) هو حقل رقم الجزء(Part number). والحقل رقم (٢) هو حقل الكمية الواردة وهو حقل عددى مكون من سبعة أرقام متضمنة رقمين عشريين. والحقل رقم (٣) هو حقل ثمن الشراء وهو حقل عددى مكون من تسعة أرقام متضمنة رقمين عشريين. والحقل رقم (٤) هو حقل تاريخ وصول الصنف وهو حقل تاريخى مكون من ثمانية حروف. والحقل رقم (٥) هو حقل إسم البائع (Vendor) القائم بالتوريد وهو حقل حرفى مكون من خمسين حرفا. والحقل رقم (٦) هو حقل الترحيل وهو حقل منطقى (Logical) يستخدم للتحكم فى ترحيل بيانات الصنف إلى الملف الرئيسى وضمان عدم تكرار ترحيلها.

ولإنشاء ملف الفهرس الخاص بهذا الملف يتم كتابة السطرين التاليين من مشيرة النقطة (Dot Prompt).

```
USE Newstock
INDEX ON Part_no TO Newstock
```

وهكذا يتكون النظام حتى الآن من ستة ملفات. ثلاثة منها ملفات قواعد بيانات (Database files) والثلاثة الآخرين ملفات فهرس (Index files).

١٢ - ٦ تصميم البرنامج

بعد تصميم ملفات قواعد البيانات كما سبق الإيضاح يأتى دور البرنامج وهو فى الواقع ليس برنامجا واحدا ولكنه عدة برامج كما سيتم الإيضاح. وهذه البرامج تهدف إلى السيطرة على ثلاثة نظم مستقلة ولكنها مرتبطة فيما بينها. النظام الأول هو نظام إدارة المخزون الحالى أو الفعلى والذى يكون مسئولا عن الموقف الفعلى للأصناف الموجودة بالمخزن ومتابعة حالتها وتحديد الأصناف التى تصل إلى حد الطلب (Reorder Point) وهكذا. والنظام الثانى هو حركة المبيعات التى يتم إدخالها بواسطة الموظف المختص فى نقطة البيع (Point of sale). والنظام الثالث هو حركة الإضافة (Newstock) التى يتم فيها تسجيل الأصناف الواردة إلى المخزن بواسطة الموظف المختص.

لذلك فإن البرنامج يمكن تقسيمه بصفة مبدئية إلى أربعة برامج أحدها برنامج رئيسى يقوم بتشغيل البرامج الثلاثة الأخرى والبرامج الثلاثة الأخرى تقوم بتشغيل برامج فرعية أخرى سيتم دراستها فيما بعد. ولكن سيتم التركيز فى هذه المرحلة على هذه البرامج الأربعة والتى تتضح من الشكل (١٢ - ٢).

ويلاحظ من الشكل أن البرنامج الرئيسى (IMenu.prg) هو البرنامج الذى يحتوى على القائمة الرئيسية التى يتم من خلالها تشغيل البرامج الأخرى. وبناء على اختيار المستخدم يتم التفرع إلى البرنامج (MMenu.prg) الذى يشغل الملف الرئيسى للأصناف (Master File) لمتابعة الأصناف الموجودة فى المخزن. أو يتم التفرع إلى البرنامج (SMenu.prg) الذى يشغل ملف المبيعات (Sales File) لمتابعة موقف الأصناف التى يتم بيعها. أو يتم التفرع إلى البرنامج (NMenu.prg) الذى يشغل ملف الإضافة (Newstock file) لمتابعة موقف الأصناف التى يتم توريدها.

IMenu.prg

1- Manage the Master Inventory
2- Manage Sales Information
3- Manage New Stock Information
4- Exit

MMenu.prg

Add New Part Numbers
Print Reports
Make Changes
Update the Master File

SMenu.prg

Record Sales
Print Reports
Make Changes

NMenu.prg

Record New Items
Print Reports
Make Changes

شكل (١٢ - ٢)

الفصل الثالث عشر

برنامج القائمة الرئيسية

برنامج القائمة الرئيسية (IMenu.prg) هو البرنامج الذى يقوم بتشغيل القائمة الرئيسية التى يختار المستخدم من خلالها أحد النظم الثلاثة كما سبق الإيضاح. وهو لايختلف فى تركيبه عن أى برنامج رئيسى يتم من خلاله عرض قائمة اختيارات (Menu choices). وكالعادة قبل كتابة أى برنامج يفضل كتابة الخطوات الأولية (PSEUDOCODE) بأى لغة يجيدها مخطط البرامج حتى يمكن بعد ذلك كتابة الأوامر بلغة برامج عائلة (DBase) التى تنفذ هذه الخطوات.

١٣ - ١ كتابة الخطوات الأولية (PSEUDOCODE)

يقوم البرنامج الرئيسى فى البداية بإنشاء متغير ذاكرة تاريخى إسمه (T_date) ويتم تخزين تاريخ اليوم الحالى (DATE()) فى هذا المتغير. ثم يتم عرض هذا التاريخ وسؤال المستخدم إذا كان هذا هو التاريخ السليم أم لا. وفى حالة اختلافه يقوم المستخدم بتعديل التاريخ. وهذه الخطوة مهمة جدا لأن هذا المتغير (T_date) سيتم استخدامه فى مواضع متفرقة داخل البرنامج. ثم يقوم البرنامج بسؤال المستخدم عن الإختيار المطلوب من القائمة التى تظهر أمامه. وبناء على اختيار المستخدم يتم التفرع إلى البرنامج الذى يحقق هذا الاختيار.

والخطوات الأولية للبرنامج (PSEUDOCODE) يتم كتابتها كالآتى :

١ - مسح كل متغيرات الذاكرة.
٢ - تجهيز بيئة البرنامج (environment).
٣ - مسح الشاشة.
٤ - إنشاء متغير ذاكرة لتاريخ اليوم الحالى.
٥ - تكوين حلقة تكرارية لعرض القائمة الرئيسية للبرنامج.
٦ - مسح الشاشة وعرض القائمة الرئيسية.
٧ - إستقبال اختيار المستخدم.
٨ - التفرع إلى البرنامج المطلوب.
٩ - إستمرار الحلقة التكرارية حتى يختار المستخدم الخروج.
١٠- الخروج من برنامج (DBase III+).

١٣ - ٢ كتابة البرنامج

يتم كتابة البرنامج كما سبق الإيضاح بكتابة (MODIFY COMMAND) يليه إسم البرنامج وهو (IMenu). وغير مطلوب إضافة الإمتداد فى هذه الحالة لأن البرنامج يضيف الإمتداد (.prg) آليا والبرنامج يتيح استخدام الحروف الأربعة الأولى فقط من كل أمر. فمثلا يمكن كتابة (MODI COMM) وهذا يوفر وقتا كبيرا عند تعديل البرنامج عدة مرات. ويتم كتابة أوامر البرنامج كالآتى :

```
************************************ IMenu.prg
*  Main menu for the inventory system
*
SET STATUS OFF
CLEAR ALL
SET TALK OFF
SET BELL OFF
SET SAFETY OFF
SET HEADING OFF
CLEAR

* - - - - - - Create underline variable Uline
Uline = REPLICATE ("_" , 80)

* - - - - - - Create memory variable for today's date
T_Date = DATE()
@ 17,5 SAY "To change date type new date and press" +;
   "Return "
@ 15,5 SAY "Today's date =" GET T_Date PICT "99/99/99"
READ

* - - - - - - Set up loop for presenting main menu.
IChoice = 0
DO WHILE IChoice # 4
```

```
CLEAR
@ 2,1 SAY 'Inventory system main menu'
@ 2,60 SAY DTOC(T_Date) + "    " + TIME()
@ 3,0 SAY Uline
?
?
TEXT
   1. Manage master inventory
   2. Record sales
   3. Record new stock
   4. Exit
ENDTEXT

* - - - - -  Wait for answer
@ 24,1 SAY "Enter choice : " GET IChoice PICT "9" ;
RANGE 1,4
READ
DO CASE
   CASE IChoice = 1
      DO MMenu
   CASE IChoice = 2
      DO SMenu
   CASE IChoice = 3
      DO NMenu
ENDCASE
ENDDO(while IChoice # 4)
* - - - - -  When done , exit
CLEAR
QUIT
```

والبرنامج يبدأ كالعادة بكتابة إسم البرنامج ووظيفته ثم يتم تجهيز بيئة البرنامج عن طريق مجموعة من أوامر (SET) وهى كالآتى :

```
SET STATUS OFF
CLEAR ALL
SET TALK OFF
SET BELL OFF
SET SAFETY OFF
SET HEADING OFF
CLEAR
```

وللتعرف على وظيفة كل من هذه الأوامر يمكن الرجوع إلى الكتاب رقم (٥) من مجموعة كتب " دلتا ".

والجزء الثانى من البرنامج يبدأ بإنشاء متغير ذاكرة (Uline) يحتوى على حرف الشرطة السفلية (_) مكررا ثمانين مرة. وهذا يؤدى إلى تخزين سطر بعرض الشاشة يمكن عرضه فى أى مكان بعد ذلك. مع ملاحظة أن إنشاء هذا المتغير فى البرنامج الرئيسى يجعله عاما (Public) بالنسبة للبرامج الفرعية الأخرى أى يمكن استخدامه مباشرة فى أى برنامج فرعى. وهذا الأمر يظهر فى البرنامج كالآتى :

```
Uline = REPLEICTE ("_" , 80 )
```

والجزء الثالث من البرنامج يبدأ بإنشاء متغير الذاكرة التاريخى (T_DATE) وتخزين تاريخ اليوم الحالى فيه. وهو التاريخ الذى يتم إدخاله عند تشغيل الجهاز من خلال نظام التشغيل (MS-DOS). وإذا لم يكن المستخدم قد أدخل التاريخ فى بداية تشغيل الجهاز فإن البرنامج يعرض له التاريخ السابق تخزينه فى المتغير (T_Date). ويستطيع فى هذه الحالة تعديل هذا التاريخ ليوافق تاريخ اليوم الحالى. وهذه العملية تتم من خلال مجموعة السطور التالية :

```
T_Date = DATE()
@ 17,5 SAY "To change date type new date and" + ;
"press Return "
@ 15,5 SAY "Today's date =" GET T_Date PICT "99/99/99"
```

والمتغير (T_date) مهم جدا لأنه سوف يستخدم فى ملء بيانات الحقول التاريخية

فى ملفات المبيعات والإضافة آليا.

والجزء الرابع يقوم بتكوين الحلقة التكرارية لمسح الشاشة وعرض العنوان متضمنا التاريخ والوقت ثم خط أسفل هذا العنوان ثم عرض القائمة الرئيسية السابق شرحها. ثم يتم عرض سؤال للمستخدم عن الإختيار المطلوب واستقبال هذا الإختيار فى المتغير (ICoice). وهذا يتم من خلال السطور التالية :

```
IChoice = 0
DO WHILE IChoice # 4
   CLEAR
   @ 2,1 SAY "Inventory system main menu"
   @ 2,60 SAY DTOC(T_Date) + "    " + TIME()
   @ 3,0 SAY Uline
   ?
   ?
   TEXT
      1. Manage master inventory
      2. Record sales
      3. Record new stock
      4. Exit
   ENDTEXT
   * - - - - - Wait for answer
   @ 24,1 SAY "Enter choice:" GET IChoice PICT "9" ;
      RANGE 1,4
   READ
```

والجزء الخامس من البرنامج يستعمل الأمر (DO CASE) فى التفرع إلى البرنامج المطلوب حسب اختيار المستخدم. ويتم ذلك من خلال السطور التالية :

```
   DO CASE
      CASE IChoice = 1
         DO MMENU
      CASE IChoice = 2
         DO SMenu
```

```
    CASE IChoice = 3
       DO NMenu
  ENDCASE
```

وفى الجزء الأخير من البرنامج يتم إنهاء الحلقة التكرارية ومسح الشاشة والرجوع إلى نظام التشغيل. وذلك كالآتى :

```
ENDDO(while IChoice # 4)
* - - - - -  When done , exit
CLEAR
QUIT
```

١٣ - ٣ إختبار البرنامج

بعد الإنتهاء من كتابة البرنامج يتم اختباره وذلك بتشغيله كالآتى :

```
DO IMenu
```

ثم ملاحظة تنفيذ خطوات البرنامج وظهور الإختيارات على الشاشة. ويمكن إدخال قيم خارج المدى المسموح وهو من (1) إلى (4) وذلك بإدخال (5) مثلا أو أى رقم آخر أكبر من (4) وملاحظة ما يحدث. وبعد انتهاء اختبار البرنامج يتم العودة إلى نظام التشغيل عن طريق الاختيار (4).

والشكل (١٣ - ١) يوضح شكل الشاشة التى تظهر عند تنفيذ هذا البرنامج.

```
Inventory system main menu    02/20/90    12:40:50

        1. Manage master inventory
        2. Record sales
        3. Record new stock
        4. Exit
    Enter choice
```

شكل (١٣ - ١)

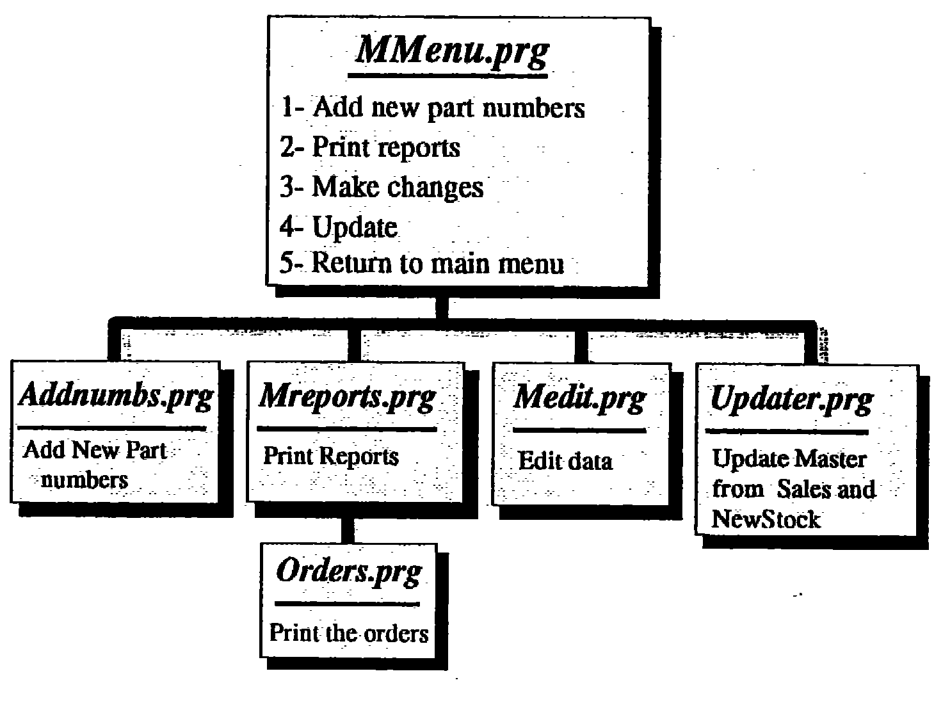
الفصل الرابع عشر

برنامج تشغيل الملف الرئيسى

هذا البرنامج يقوم بإدارة وتشغيل المعلومات المخزنة فى الملف الرئيسى (Master file) بهدف تحديد الحالة الفعلية (Current Status) للأصناف فى المخزن. وهذا الملف سبق تكوينه وتسميته (Master.dbf). كما سبق إنشاء الفهرس الخاص به بناء على حقل رقم الجزء (Part_no). ويتم تشغيل هذا البرنامج عند اختيار الرقم (١) من القائمة الرئيسية. إرجع إلى الشكل (١٣ - ١)

١٤ - ١ تصميم برنامج تشغيل الملف الرئيسى

يتكون برنامج تشغيل الملف الرئيسى من ستة برامج منفصلة. أحدها يمثل البرنامج الرئيسى (Main Program) الذى يتحكم فى البرامج الخمسة الأخرى أنظر الشكل (١٤ - ١).



شكل (١٤ - ١)

ويوضح الشكل إسم كل برنامج ثم وظيفة هذا البرنامج مع ملاحظة أن البرنامج الرئيسى (MMenu.prg) وظيفته الرئيسية عرض القائمة المبينة والتفرع إلى كل برنامج من البرامج الفرعية بناء على اختيار المستخدم.

١٤ - ٢ تصميم البرنامج الرئيسى

البرنامج الرئيسى فى هذه الحالة لايختلف عن برامج القائمة الرئيسية السابق شرحها. لذلك ليست هناك حاجة لشرحه بالتفصيل ونكتفى هنا بكتابة البرنامج كالآتى :

```
* * * * * * * * * * * * * * * MMenu.prg
*   Menu for master program of inventory system
*   Called from inventory system main menu
* - - - - -  Set up loop for presenting menu
Mchoice = 0
DO WHILE Mchoice # 5
      CLEAR
      @ 2,1  SAY "Manage Master Inventory"
      @ 2,60 SAY DTOC(T_Date) + "   " + TIME()
      @ 3,0  SAY Uline
      ?
      ?
      TEXT

            1. Add new part numbers
            2. Print reports
            3. Make changes
            4. Update from Sales and Newstock
            5. Return

      ENDTEXT
      @ 24,1 SAY "Enter choice(1-5)" GET Mchoice PICT "9";
      RANGE 1,5
      READ
      DO CASE
            CASE Mchoice = 1
                  DO Addnumbs
```

```
        CASE Mchoice = 2
            DO Mreports
        CASE Mchoice = 3
            DO Medit
        CASE Mchoice = 4
            DO Updater
        ENDCASE
ENDDO(While Mchoice # 5)

* - - - - -  when done , return to main menu.
RETURN
```

وعند تنفيذ هذا البرنامج تظهر الشاشة الموضحة فى الشكل (١٤ - ٢)

```
Manage Master Inventory          02/20/90     12:40:50

        1.  Add new part numbers
        2.  Print reports
        3.  Make changes
        4.  Update from Sales and Newstock
        5.  Return to main menu
  Enter  choice  (1 - 5)
```

شكل (١٤ - ٢)

١٤ - ٣ برنامج إضافة الأصناف

هذا البرنامج هو البرنامج الذى يتم التفرع إليه عند اختيار المستخدم للرقم (١) فى القائمة السابقة. فى هذه الحالة يتم تنفيذ البرنامج (Addnumbs.prg). وقبل كتابة البرنامج يجب أولا إنشاء شاشة الإدخال التى سوف تستخدم فى إدخال بيانات الصنف الجديد.

١٤ - ٣ - ١ إنشــاء شــاشــة الإدخال

يتم إنشاء شاشة الإدخال بإحدى طريقتين ، الطريقة الأولى باستخدام راسم الشاشة (Screen Painter) الذى يظهر عن طريق قوائم المساعد (Assistant) أو عن طريق كتابة الأمر (CREATE SCREEN) من خلال مشيرة النقطة (Dot Prompt). والطريقة الثانية عن طريق كتابة ملف أوامر (Command File). وسنستخدم الطريقة الأولى فى هذا البرنامج.

ولتنفيذ ذلك يتم كتابة السطر التالى عند مشيرة النقطة (Dot Prompt) :

CREATE SCREEN Iscreen1

وعند ظهور عمود الإختيارات على الشاشة يتم اختيار (Database File). ومن خلاله يتم اختيار ملف قاعدة البيانات (Master.dbf) ثم يتم اختيار (Load Fields) حيث يتم اختيار الحقول المطلوب ظهورها فى الشاشة. ويتم اختيار كل حقل عن طريق تحريك العمود الضوئى إلى هذا الحقل والضغط على مفتاح الإدخال. فى هذه الحالة يظهر مثلث ( ) أمام إسم الحقل. ويمكن اختيار كل الحقول ما عدا حقل تاريخ آخر طلب للصنف (Order_Date) وحقل الكمية التى يجب طلبها من الصنف (New_Order) لأن هذين الحقلين يتم ملؤهما آليا من خلال البرنامج. وعند الإنتهاء من إدخال الحقول يتم الضغط على مفتاح (<--) للخروج من هذه القائمة ولعرض السبورة (Blackboard) التى سيتم من خلالها تحديد مواضع الحقول حسب الحاجة.

وعند الإنتهاء من تحديد مواضع الحقول على الشاشة يتم تحريك المؤشر إلى العمود الضوئى الخاص برقم الجزء (Part number) ويتم تحويل حالة هذا الحقل من (Edit Get) إلى (Display Say). وذلك لكى يصبح هذا الحقل غير قابل للتعديل بواسطة المستخدم.

ولتنفيذ ذلك يتم الضغط على مفتاح (F10) والضغط على مفتاح الإدخال عند الاختيار (ACTION) لتحويله إلى (DISPALY SAY). ويلاحظ فى هذه الحالة اختفاء العمود الضوئى الخاص برقم الجزء من الشاشة.

ولمزيد من التفاصيل عن تصميم شاشة الإدخال يمكن الرجوع إلى الكتاب رقم (٥) من مجموعة كتب " دلتا ". والشكل رقم (١٤ - ٣) يوضح تصميم مقترح لشاشة الإدخال ويمكن تصميم أى شكل آخر حسب الحاجة.

Master Inventory File

Part number :
Date :
Part name :
Unit cost :
Quantity in stock :
Reorder Point :
Quantity on order :
Storage location :
Vendor :
Name
Address

شكل (١٤ - ٣)

١٤ - ٣ - ٢ كتابة الخطوات الأولية (PSEUDOCODE)

بعد تصميم الشاشة يتم كتابة الخطوات الأولية لبرنامج(Addnumbs.prg) وذلك كالآتى :

١- يتم فتح الملف الرئيسى (Master.dbf) وملف الفهرس الخاص به.
٢- يتم تكوين حلقة تكرارية لإضافة الأصناف الجديدة.
٣- يتم مسح الشاشة.
٤- يتم سؤال المستخدم عن رقم الصنف المطلوب إضافته.
٥- يتم اختبار هذا الرقم للتأكد أنه لم يسبق إدخاله فى الملف.
٦- إذا لم يتم إدخال رقم الصنف يتم الرجوع إلى القائمة الرئيسية.
٧- إذا كان الرقم موجودا يتم تنبيه المستخدم لإعادة المحاولة مرة ثانية.

٨- إذا كان الرقم غير موجود يتم إضافته وفتح شاشة الإدخال لإدخال بيانات هذا الصنف الجديد.

٩- يتم استمرار تنفيذ الحلقة التكرارية حتى يضغط المستخدم على مفتاح الإدخال دون إدخال رقم صنف جديد.

١٠- يتم الرجوع إلى القائمة الرئيسية.

١٤ - ٣ - ٣ كتابة برنامج الإضافة (Addnumbs.prg)

يتم كتابة الأمر (MODI COMM Addnumbs) من مشيرة النقطة ثم يتم كتابة سطور البرنامج كالآتى :

```
* * * * * * * * * * * * * * *Addnumbs.prg
*    Add new items to the Master file
*    Called from MMenu.prg

USE Master INDEX Master

* - Set up loop for adding new part numbers
Partnumb = "X"
DO WHILE Partnumb # " "
        CLEAR
        @ 2, 1 SAY "Add New Part Numbers"
        @ 2,60 SAY DTOC(T_Date) + "   " + TIME()
        @ 3, 0 SAY Uline
        ?
        ?
        * - - - - - GET proposed part number
        Partnumb = SPACE(6)
        @ 15,6 SAY "Enter part number(or press Return to exit)" ;
             GET Partnumb
        READ
```

```
* - Check to see if part number already exists
partnumb = UPPER(partnumb)
SEEK partnumb
DO CASE
    CASE Partnumb = "  "
        CLEAR
    CASE FOUND()
        @ 20,10 SAY Partnumb + "already exists"
        ? CHR(7)
        WAIT 'Press any key to try again'
    CASE .NOT. FOUND()
        APPEND BLANK
        REPLACE Part_no WITH Partnumb
        REPLACE Date WITH T_Date
        SET FORMAT TO Iscreen1
        READ
        SET FORMAT TO
ENDCASE
ENDDO(While partnumb = "  " )
* - - - - - Return to Master menu
RETURN
```

والجزء الأول من البرنامج يبدأ كالعادة بكتابة إسم البرنامج ووظيفته ثم تحديد البرنامج القائم باستدعائه. بعد ذلك يتم فتح ملف قاعدة البيانات (Master.dbf) وملف الفهرس المقابل (Master.ndx).

والجزء الثانى من البرنامج يتم من خلاله تكوين الحلقة التكرارية بعد إنشاء المتغير الحرفى (Partnumb) وإعطائه القيمة (X). والقيمة (X) فى هذه الحالة تضمن تنفيذ الحلقة التكرارية على الأقل مرة واحدة. ثم يبدأ تنفيذ الحلقة التكرارية بمسح الشاشة وعرض عنوان (Header) مع عرض التاريخ والوقت ثم يتم سؤال المستخدم عن رقم الجزء المطلوب إضافته وتخزين هذا الرقم فى المتغير(Partnumb). ويجب ملاحظة أن الضغط على مفتاح الإدخال دون

كتابة رقم الجزء يؤدى إلى الخروج من الحلقة التكرارية والرجوع إلى القائمة الرئيسية. وهذا الجزء يتكون من السطور التالية :

```
Partnumb = "X"
DO WHILE Partnumb # " "
      CLEAR
      @ 2, 1 SAY "Add New Part Numbers"
      @ 2,60 SAY DTOC(T_Date) + "   " + TIME()
      @ 3, 0 SAY Uline
      ?
      ?
      * - - - - -  GET proposed part number
      Partnumb = SPACE(6)
      @ 15,6 SAY "Enter partnumber(or press  Return to exit)" ;
           GET Partnumb
      READ
```

والجزء الثالث من البرنامج يبدأ بتحويل رقم الجزء إلى حروف كبيرة (Uppercase) ثم البحث عن رقم الجزء فى الملف الرئيسى. وحيث أن الملف مفهرس لذلك يستخدم الأمر (SEEK) فى البحث عن هذا الرقم.

ويحتوى هذا الجزء على السطور التالية :

```
Partnumb = UPPER(partnumb)
SEEK Partnumb
```

والجزء الرابع من البرنامج يختص باتخاذ القرار بناء على ما يدخله المستخدم. وفى الحالة الأولى عندما يكون (" " = Partumb) فإن هذا يعنى أن المستخدم يضغط على مفتاح الإدخال دون كتابة أى رقم جزء (Partnumb). وفى هذه الحالة يتم مسح الشاشة ثم تنفيذ الأمر الذى يلى الأمر (ENDCASE). وهذا يؤدى إلى الخروج من الحلقة التكرارية لأن الشرط الموجود فى أول الحلقة التكرارية لايتحقق.

وفى الحالة الثانية (CASE FOUND()) ، فإن هذا يعنى أن هذا الجزء قد سبق إدخاله فى الملف. لذلك يتم تنبيه المستخدم بعرض رسالة توضح له أن هذا الرقم موجود. وذلك بالإضافة إلى تشغيل الجرس للتنبيه مع استخدام الأمر (WAIT) فى عرض الرسالة الموضحة وانتظار ضغط المستخدم على أى مفتاح حتى يمكنه إدخال رقم جديد.

وفى الحالة الثالثة (CASE .NOT. FOUND()) فإن هذا يعنى أن المستخدم أدخل رقما غير موجود داخل الملف. وفى هذه الحالة فإن البرنامج يعرض شاشة الإدخال ثم يسمح للمستخدم بإدخال باقى بيانات الصنف مع عدم السماح له بكتابة رقم الجزء من خلال شاشة الإدخال. وإنما يتم إدخال رقم الجزء بواسطة البرنامج عن طريق الأمر (REPLACE). كما يتم إدخال تاريخ آخر تعديل (Date) عن طريق البرنامج أيضا.

ويتكون هذا الجزء من السطور التالية :

```
DO CASE
    CASE Partnumb = "  "
        CLEAR
    CASE FOUND()
        @ 20,10 SAY Partnumb + "already exists"
        ? CHR(7)
        WAIT 'Press any key to try again"
    CASE .NOT. FOUND()
        APPEND BLANK
        REPLACE Part_no WITH Partnumb
        REPLACE Date WITH T_Date
        SET FORMAT TO Iscreen1
        READ
        SET FORMAT TO
ENDCASE
```

والجزء الأخير من البرنامج يحتوى على أمر إنهاء الحلقة التكرارية والعودة إلى القائمة الرئيسية الخاصة بالملف الرئيسى (MMenu). وإذا لم يرد المستخدم الخروج فإن الحلقة التكرارية تستمر. أما إذا أراد الخروج فإنه يضغط على مفتاح الإدخال دون إدخال أى رقم وفى هذه الحالة يتم الرجوع إلى القائمة الرئيسية.

١٤ - ٤ برنامج تقارير الملف الرئيسى

يمكن تصميم تقارير الملف الرئيسى (Master.dbf) عن طريق قوائم برنامج المساعد (Assistant) أو باستخدام الأمر (CREATE REPORT). وفى الحالتين تظهر اختيارات تصميم التقرير التى يتم عن طريقها إدخال المعاملات المختلفة التى توضح عنوان الصفحة (Page Title) وأسماء الحقول بالإضافة إلى المعاملات الأخرى. وكما سبق الإيضاح فإن هناك طريقة أخرى لتصميم التقرير عن طريق ملف الأوامر (Command file). وهى تتيح إمكانيات أكبر فى تحديد أماكن الحقول وشكل التقرير بصفة عامة. وبالنسبة للملف الرئيسى (Master.dbf) فسوف نتعرض للطريقتين عند إنشاء التقارير الخاصة به.

وهناك أربعة تقارير مطلوبة للملف الرئيسى يتم شرحها فى الأجزاء التالية :

١٤ - ٤ - ١ تقرير المخزون الحالى (Current Stock)

هذا التقرير يوضح الموقف الحالى للأصناف فى المخزن من حيث إسم كل صنف والكمية الفعلية الموجودة منه ومكانه فى المخزن وسعر الوحدة وهكذا. ويتم إنشاء هذا التقرير عن طريق كتابة السطرين التاليين من مشيرة النقطة (Dot Prompt).

```
USE Master
CREATE REPORT Allmast
```

حيث (Allmast) هو إسم ملف التقرير. ويلاحظ فى هذه الحالة ظهور قوائم تصميم التقرير السابق شرحها فى الكتاب رقم (٥) من مجموعة كتب " دلتا " ولن نتعرض لها هنا بالتفصيل ولكن سيتم توضيح المعاملات التى يجب إدخالها حتى نحصل على شكل التقرير المطلوب. والشكل التالى يوضح قيم المعاملات الخاصة بصفحة التقرير.

Page Title	Current Stock
Page width	74
Left margin	1
Right margin	0
Lines per page	52
Double space report	NO

شكل (١٤ - ٤)

كما يتم تحديد محتويات أعمدة التقرير (Columns) كالآتى :

Column No.	Contents	Heading	Width	Decimals	Total
1	Part_no	Part No	6		
2	Title	Part Name	20		
3	Qty	On Hand	7	2	N
4	Cost	Unit Cost	9	2	N
5	Reorder	Reorder	8	2	N
6	Location	Location	9		
7	Date	Last Update	9		

شكل (١٤ - ٥)

ولتوضيح شكل التقرير بعد طباعته نفرض بيانات بعض الأصناف. ثم نكتب الأمر (REPORT FORM Allmast). وفى هذه الحالة يظهر التقرير كالآتى مثلا :

Page No 02/20/90						Current Stock
Part No.	Part Name	On Hand	Unit Cost	Reorder	Location	Last Update
BBB	Floopy Disk		40	14	10-A-111	02/15/90
AAA	Printer		10	800	17-B-233	02/10/90
ZZZ	Bicycle		10	80	19-C-175	01/01/90

شكل (١٤ - ٦)

١٤ - ٤ - ٢ تقرير حد الطلب (Reorder)

وهو التقرير الذى يعرض بيانات الأصناف التى تقل عن حد الطلب. ويتم إنشاؤه عن طريق الأمر (CREATE REPORT Reorders) كما سبق الإيضاح. ويتم استخدام المعاملات الموضحة فى الشكل التالى :

Page Title	Goods to be Reordered
Page width	77
Left margin	1
Right margin	0
Lines per page	58
Double space report	NO

شكل (١٤ - ٧)

كما يتم إدخال محتويات أعمدة التقرير (Columns) كالآتى :

Column No.	Contents	Heading	Width	Decimals	Total
1	Part_no	Part No	6		
2	Title	Part Name	20		
3	Qty	On Hand	7	2	N
4	Cost	Unit Cost	9	2	N
5	Reorder	Reorder	8	0	N
6	Location	Location	9		
7	Date	Last Update	8		

شكل (١٤ - ٨)

ولتوضيح شكل التقرير عند طباعته يتم كتابة الأمر (REPORT FORM Reorders) فى هذه الحالة يظهر التقرير الآتى :

Page No 1
02/20/90

Godes to be Reordered

Part No.	Part Name	On Hand	On Order	Reorder	Vendor Name
AAA	Printer	5	5	10	Micronet Company

شكل (١٤ - ٩)

١٤ - ٤ - ٣ تقرير الأصناف تحت الطلب

ويتم انشاء هذا التقرير بنفس الطريقة مثل التقارير السابقة عن طريق الأمر (CREATE REPORT Onorder) مع ادخال المعاملات الواضحة فى الشكل التالى :

Page Title	Items Currently On Order
Page width	77
Left margin	1
Right margin	0
Lines per page	58
Double space report	NO

شكل (١٤ - ١٠)

كما يتم إدخال بيانات الأعمدة بالإستعانة بالبيانات الموجودة فى الشكل التالى :

Column No.	Contents	Heading	Width	Decimals	Total
1	Order_date	Order Date	8		
2	Part_no	Part No	6		
3	Title	Part Name	15		
4	On-order	On Order	7	2	N
5	Cost	Unit Cost	9	2	N
6	Cost *On_order	Total Cost	9	2	Y
7	Vendor	Vendor Name	25		

شكل (١٤ - ١١)

ولعرض التقرير يتم كتابة الأمر (REPORT FORM Onorder).

وفى هذه الحالة يظهر التقرير كالآتى مثلا :

Page No 1 02/22/90		Item Quantity On Order				
Order Date	Part No	Part Name	On Order	Unit Price	Total Cost	Vendor Name
01/02/90	AAA	Printer	5	800	4000	Micronet Company
TOTAL					4000	

شكل (١٤ - ١٢)

ويلاحظ هنا أن التقرير يقوم بتجميع البيانات الموجودة فى عمود التكلفة الكلية (Total Cost). وذلك لأنه قد سبق إدخال الإختيار (Y) فى العمود (Total?) كما يتضح من الشكل الخاص ببيانات الأعمدة.

١٤ - ٤ - ٤ طلب الشراء (Purchase Order)

طلب الشراء هو أحد التقارير التى يتم الحصول عليها من خلال برنامج تشغيل الملف الرئيسى (Master.dbf). والبرنامج يعرض على المستخدم بيانات الأصناف التى تصل إلى حد الطلب ويتيح للمستخدم طلب العدد الذى يريده من هذه الأصناف. حيث يعرض البرنامج الشاشة التالية لكل صنف من الأصناف التى تصل إلى حد الطلب.

Part number	D-100	Suit
On hand On order Reorder Unit cost	30 10 50 100	
Order now many ?		

شكل (١٤ - ١٣)

ومن خلال هذه الشاشة يستطيع المستخدم أن يلاحظ بسرعة عدد البدل (Suits) الموجودة فى المخزن وعدد البدل الجارى طلبها (On Order) وحد الطلب (Reorder). وبناء على ذلك يحدد عدد البدل المطلوب شراؤها لتعويض النقص الموجود. ويقوم المستخدم بإدخال هذا العدد أمام السؤال المبين. ويتم تكرار هذه العملية مع باقى الأصناف التى وصلت إلى حد الطلب (Reorder). وبعد الإنتهاء من إدخال الأعداد المطلوبة من كل صنف يقوم البرنامج آليا بطباعة طلبات الشراء لهذه الأصناف. والشكل التالى يوضح نموذجا لأحد هذه الطلبات.

Omar Afandy Company
Abbasia street

Please send us the following itmes...

10	Suits	100	1000
5	Printers	800	4000
Total Cost			5000

Mail to AAA Company
12 - Ahram Street

شكل (١٤ - ١٤)

١٤ - ٤ - ٥ تصميم برنامج التقارير

يتكون برنامج التقارير من برنامجين أحدهما هو البرنامج (MReports.prg) الذى يعرض القائمة الرئيسية للتقارير وهى القائمة التى يختار المستخدم منها التقرير المطلوب طباعته. والبرنامج الآخر هو برنامج طلبات الشراء (Orders.prg) الذى يتم عن طريقه طباعة طلبات الشراء للأصناف التى تقل عن حد الطلب فى المخزن.

١٤ - ٤ - ٦ تصميم برنامج القائمة

وهذا البرنامج لايختلف عن برامج القوائم الأخرى. ولكن يجب ملاحظة أن

فتح ملف قاعدة البيانات وملف الفهرس المرتبط به يتم فى بداية البرنامج وقبل الدخول فى الحلقة التكرارية. وذلك لأن الملفين يستخدمان فى جميع التقارير بعد ذلك. والبرنامج يتكون من السطور التالية :

```
* * * * * * * * * * * * * * * MRoports.prg
*   Present report options for Master file
*   Called from Master menu , MMenu.prg
USE Master INDEX Master
* - - - - - Set up loop for presenting menu
Repchoice = 0
DO WHILE Repchoice # 5
        CLEAR
        @ 2,1  SAY "Master Inventory Report Options"
        @ 2,60 SAY DTOC(T_Date) + "   " + TIME()
        @ 3,0  SAY Uline
        ?
        ?
        TEXT

                1. Entire inventory
                2. Reorder report
                3. On Order report
                4. Purchase orders
                5. Return to master menu

        ENDTEXT
        @ 24,1 SAY "Enter choice(1-5)" ;
            GET Repchoice PICT "9" RANGE 1,5
        READ
        * - - If not choosing Purchase orders, ask about
        * - - -  printer .
        CLEAR
        STORE " " TO YN, Printer
```

```
    IF Repchoice < 4
        @ 5,5 SAY "Send report to printer ?" GET YN  PICT "!"
        READ
        CLEAR

        * - - - - - Set up for printer.
        IF YN = "Y"
            Printer = "TO PRINT"
        ENDIF
    ENDIF(Repchoice < 4)
  DO CASE
    CASE Repchoice = 1
        REPORT FORM Allmast & Printer
    CASE Repchoice = 2
        REPORT FORM Reorders FOR (QTY + On_order) ;
        < = Reorder & Printer
    CASE Repchoice = 3
        REPORT FORM Onorder FOR On_order > 0  & Printer
    CASE Repchoice = 4
        DO Orders
  ENDCASE

  * - - - IF report not going to printer and not
  * - - - exiting program , pause .
  IF YN # "Y" .AND. Repchoice # 5
    ?
    ?
    WAIT "Press any key to return to the reports menu"
  ENDIF
ENDDO (Repchoice # 5)
RETURN
```

والجزء الأول من البرنامج يبدأ بالسطور المعتادة لكتابة إسم البرنامج الذى يقوم باستدعائه. ثم يتم فتح الملف الرئيسى (Master.dbf) وملف الفهرس الخاص به (Master.ndx). ويتم بعد ذلك تكوين الحلقة التكرارية التى يتم من خلالها عرض قائمة الإختيارات التى يختار المستخدم منها نوع التقرير المطلوب. ويتكون هذا الجزء من السطور التالية :

```
USE Master INDEX Master
Repchoice = 0
DO WHILE Repchoice # 5
CLEAR
   @ 2,1  SAY "Master Inventory Report Options"
   @ 2,60 SAY DTOC(T_Date) + "   " + TIME()
   @ 3,0  SAY Uline
   ?
   ?
   TEXT
          1. Entire inventory
          2. Reorder report
          3. On Order report
          4. Purchase orders
          5. Return to master menu
   ENDTEXT
   @ 24,1 SAY "Enter choice(1-5)" ;
     GET Repchoice PICT "9" RANGE 1,5
   READ
```

والاختيارات الثلاثة الأولى تستخدم التقارير التى سبق إنشاؤها فى عرض البيانات المطلوبة. أما الاختيار رقم (4) فإنه يؤدى إلى تشغيل البرنامج (Orders.prg). وهذا البرنامج يعرض على المستخدم بيانات الأصناف التى تقل عن حد الطلب (Reorder) ويتيح له تحديد الكمية المطلوبة من كل صنف. ثم يقوم البرنامج بطباعة أوامر الشراء آليا.

والجزء الثانى يبدأ بإنشاء متغيرات الذاكرة (YN) و (Printer) وإعطائهما القيمة (" "). ثم سؤال المستخدم إذا كان يريد طباعة التقرير على الطابعة أو عرضه على الشاشة وذلك بالنسبة للإختيارات التى تقل عن (4) وذلك لأن الاختيار (4) يتفرع إلى برنامج (Orders.prg) الذى يقوم بطباعة أوامر الشراء (Purchase orders).

وعندما يختار المستخدم الاختيار (Y) فإن الشرط الموجود بعد (IF) يتحقق. ويؤدى هذا إلى تخزين العبارة (TO PRINT) فى المتغير (Printer). ويتكون هذا الجزء من السطور التالية :

```
CLEAR
STORE " " TO YN, Printer
IF Repchoice < 4
   @ 5,5  SAY "Send report to printer ?" ;
     GET YN PICT "!"
   READ
   CLEAR

   * - - - - -  Set up for printer.
   IF YN = "Y"
      Printer = "TO PRINT"
   ENDIF
ENDIF(Repchoice < 4)
```

والجزء الثالث يتم فيه اتخاذ القرار تبعا لاختيار المستخدم. فعندما يختار الرقم (1) يتم تكوين التقرير (Allmast) الذى سبق إنشاؤه. ويلاحظ استخدام دالة الماكرو (&) للتعويض عن المتغير (Printer) بالقيمة المخزنة فيه وهى ("TO PRINT"). أى أن الأمر يصبح كالآتى :

```
REPORT FORM Almast TO PRINT
```

وهذا يؤدى إلى طباعة التقرير متضمنا بيانات الأصناف الموجودة بالمخزن.

وعندما يختار المستخدم الرقم (2) يتم تكوين التقرير (Reorders) الذى سبق إنشاؤه. ويلاحظ أن الأمر المستخدم فى هذه الحالة يستخدم شرطا لتحديد السجلات التى تظهر فى التقرير. وهى سجلات الأصناف التى تقل كميتها عن حد الطلب.

وعندما يختار المستخدم الرقم (3) يتم طباعة التقرير (Onorders) بالنسبة للأصناف التى توجد منها كميات تحت الطلب فقط وذلك عن طريق استخدام الشرط (On_order >0).

واستخدام دالة الماكرو (&) فى الإختيارات الثلاث السابقة يفيد فى التحكم فى طباعة التقرير أو عرضه على الشاشة حسب اختيار المستخدم. فإذا أراد المستخدم طباعة التقرير فإنه يدخل الحرف (Y) فى المتغير (YN). وهذا يؤدى إلى إدخال العبارة ("TO PRINT") فى المتغير (printer) كما سبق الإيضاح. وهذا بالتالى يؤدى إلى إدخال عبارة (TO PRINT) بعد الأمر (REPORT FORM) مما يؤدى إلى طباعة التقرير.

أما إذا أراد المستخدم عرض التقرير على الشاشة فقط دون طباعته فإنه يدخل أى قيمة أخرى غير (Y) فى المتغير (YN). وبالتالى لايتحقق الشرط بعد (IF) ويظل المتغير (printer) خاليا. وهذا يؤدى إلى عدم إضافة أى عبارة بعد الأمر (REPORT FORM) وبالتالى يتم عرضه على الشاشة فقط.

وعندما يختار المستخدم الإختيار (4) فإن البرنامج يتفرع إلى البرنامج الفرعى (Orders) الذى سيتم دراسته فيما بعد.

والجزء الثالث الذى سبق شرحه يتكون من السطور التالية :

```
DO CASE
        CASE Repchoice = 1
                REPORT FORM Allmast & Printer
        CASE Repchoice = 2
                REPORT FORM Reorders FOR (QTY + On_order) ;
                        < = Reorder & Printer
```

```
        CASE Repchoice = 3
                REPORT FORM Onorder FOR On_order > 0 ;
                        & Printer
        CASE Repchoice = 4
                DO Orders
ENDCASE
```

والجزء الرابع والأخير من البرنامج يتم من خلاله إيقاف الشاشة مؤقتا فى حالة اختيار المستخدم عرض التقرير على الشاشة وليس على الطابعة. وذلك حتى يستطيع المستخدم قراءة بيانات التقرير على الشاشة. ثم يقوم بالضغط على أى مفتاح للرجوع إلى قائمة التقارير مرة أخرى حتى يختار نوعا آخر من التقارير حسب الحاجة أو يختار الرقم (5) للخروج من برنامج التقارير والعودة إلى القائمة الرئيسية لبرنامج تشغيل الملف الرئيسى (MMenu.prg).

١٤ - ٤ - ٧ برنامج أوامر الشراء

لعرض أو طباعة أوامر الشراء (Purchase Orders) فإن برنامج التقارير (MReports.prg) يتفرع إلى برنامج أوامر الشراء (orders.prg). وهذا البرنامج يؤدى عدة وظائف فهو فى البداية يجب أن يبحث خلال الملف الرئيسى (Master.dbf) عن الأصناف التى يجب طلبها. ولتنفيذ ذلك فإنه يجمع الكمية الموجودة فعلا (On hand) على الكمية تحت الطلب (On order) ويقارن المجموع بحد الطلب (Reorder). وفى كل مرة يجد فيها صنفا يقل عن حد الطلب فإنه يسأل المستخدم عن الكمية المطلوب شراؤها من هذا الصنف. وبعد الإنتهاء من إدخال كل الأصناف المطلوب شراؤها فإنه يطبع أوامر الشراء. وفى نفس الوقت يقوم بتحديث الكمية الموجودة فى الحقل (On_order) فى الملف الرئيسى وذلك بجمع الكمية الجديدة على هذا الحقل بالنسبة لكل صنف يتم إدخاله فى أوامر الشراء.

ولتصميم هذا البرنامج نقوم أولا بكتابة الخطوات الأولية (PSEUDOCODE) وهى تكون كالآتى :

١ - يتم مسح الشاشة.
٢ - يتم فتح ملف قاعدة البيانات وملف الفهرس الخاص به.

٣ - يتم تكوين حلقة تكرارية خلال الملف الرئيسى (Master.dbf).
٤ - يتم اختبار مجموع الكمية الفعلية (onhand) مع الكمية تحت الطلب (Onorder) ومقارنة المجموع بحد الطلب. وذلك بالنسبة لكل صنف من أصناف الملف الرئيسى (Master.dbf).
٥ - يتم مسح الشاشة.
٦ - يتم عرض بيانات حالة الصنف الذى يجب طلبه.
٧ - يتم سؤال المستخدم عن الكمية المطلوب طلبها من هذا الصنف.
٨ - يتم تحديث الحقل (New_order) الذى يمثل الكمية التى يجب طلبها من الصنف وكذلك تحديث الحقل (Order_date) الذى يمثل تاريخ آخر طلب لهذا الصنف.
٩ - عند إدخال كل الأصناف التى يجب طلبها يتم مسح الشاشة.
١٠- يتم إنشاء ملف مؤقت (Temporary File) للأصناف المطلوبة.
١١- يتم تحديث حقل تحت الطلب (On_order) فى الملف الرئيسى (Master.dbf).
١٢- يتم إعادة محتويات حقل (New_order) إلى الصفر.
١٣- يتم فتح الملف المؤقت الذى يحتوى على بيانات الأصناف المطلوبة فقط.
١٤- يتم فهرسة هذا الملف بناء على حقل البائع (Vendor) وذلك لتقسيم الأصناف بالنسبة للبائعين.
١٥- يتم تشغيل الطابعة.
١٦- يتم طباعة إسم البائع وعنوانه.
١٧- يتم طباعة الكمية المطلوبة وإسم الصنف وسعره لكل صنف من الأصناف التى يتم شراؤها من هذا البائع.
١٨- عند الانتهاء من هذا البائع يتم طباعة السعر الكلى لهذا الطلب كما يتم طباعة إسم المكان المطلوبة له هذه الأصناف وعنوانه.
١٩- يتم الإنتقال إلى صفحة جديدة على الطابعة وكذلك الإنتقال إلى بائع جديد (Vendor) فى الملف المؤقت.
٢٠- تستمر هذه العملية بالنسبة لجميع البائعين فى الملف المؤقت.
٢١- عند الإنتهاء يتم إغلاق الطابعة والعودة الى برنامج التقارير.

١٤ - ٤ - ٨ كتابة البرنامج

بعد كتابة الخطوات الأولية (PSEUDOCODE) للبرنامج يتم كتابة أوامر

البرنامج وهي تتلخص فى السطور التالية :

```
* * * * * * * * * * * * * * * * Orders.prg
*    Create Purchase orders for reordering
*    Called from Reports menu, MReports.prg
CLEAR
USE Master INDEX Master
* - -Make the loop,and display goods below reorder
* - -point,and ask the user how many of each to order.
GO TOP
DO WHILE .NOT. EOF()

        * - Find out if on_hand plus on_order quantity is
        * - - less than reorder point .
        IF (Qty + On_order) < = Reorder
                CLEAR
                * - - - - - Show status of item to be reordered
                @ 5,5  SAY "Part number: " + Part_no +" "+ P_name
                @ 6,5  SAY "On hand " + STR(Qty,4)
                @ 7,5  SAY "On order " + STR(On_order,4)
                @ 8,5  SAY "Reorder   " + STR(Reorder,3)
                @ 9,5  SAY "Unit Cost" + STR(Cost,9,2)

                * Ask user how many to order
                @ 12,5 SAY "Order how many?" GET New_order ;
                    PICT "999"
                REPLACE Order_date WITH T_Date
                READ
        ENDIF
        SKIP
ENDDO(Continue loop untill end of file)

* - - - When all orders have been placed
```

```
* - - - Make a temporary file of items to be ordered.
CLEAR
? "Preparing Files ..... please wait"
?
? "(Prepare printer while waiting)"
COPY TO Temp FOR New_order > 0

* - -Update On_order field in Master file with new
* - -orders,then set the New_order field back to zero.
REPLACE ALL On_order WITH On_order + New_order
REPLACE ALL New_order WITH 0

* - - - -  Use temp file (which contains new orders)
* - - - -  Indexed by vendor
USE Temp
INDEX ON UPPER(Vendor) TO Temp

* - - - - -  Files ready , inform user
CLEAR
? CHR(7)
WAIT "Ready printer and press any key to print orders"
SET PRINT ON
GO TOP

* - - - - -  Loop through Temp file
DO WHILE .NOT. EOF()

        * For each vendor , print name and address
        This_loop = Vendor
        Mtotal = 0
        ? vendor
        ? Vendor_add
```

```
?
?
? "Please send us the following items...."

* - -For each item to be ordered from this vendor
* - -print quantity,item , and price .
DO WHILE Vendor = this_loop .AND. .NOT. EOF()
        ? New_order , P_name , Cost , New_order * Cost
        Mtotal = New_order * Cost + Mtotal
        SKIP
ENDDO

* - - - When done with this vendor, print total
* - - - Cost,and shipping name and address .
?
? "Total cost :        ", Mtotal
?
? "Mail to : My company"
? "        12 - Tayaran street"
EJECT
ENDDO(Continue for each vendor in Temp file)
* - - - - when done, turn off printer , and return to
* - - - - Reports menu .
SET PRINT OFF
RETURN
```

والجزء الأول من البرنامج يبدأ بالتعريف بإسم البرنامج ووظيفته وإسم البرنامج الذى قام باستدعائه. ثم يقوم بمسح الشاشة وفتح الملف الرئيسى (Master.dbf) وملف الفهرس الخاص به. وذلك من خلال السطور التالية :

```
CLEAR
USE Master INDEX Master
```

والجزء الثانى من البرنامج يتم من خلاله إنشاء حلقة تكرارية (Loop) للبحث خلال الملف الرئيسى (Master.dbf) عن الأصناف التى يقل مجموع الكمية الموجودة منها والكمية تحت الطلب عن حد الطلب لهذه الأصناف. كما يتم عرض بيانات هذه الأصناف ثم يتم سؤال المستخدم عن الكمية المطلوب صرفها من كل صنف من هذه الأصناف. ويتم ذلك من خلال السطور التالية :

```
GO TOP
DO WHILE .NOT. EOF()
	* - - - Find out if on_hand plus on_order
	* - - - quantity is less than reorder point
	IF (Qty + On_order) < = Reorder
		CLEAR

		* - - - Show status of item to be reordered
		@ 5,5 SAY "Part number:" + Part_no + " "+ P_name
		@ 6,5  SAY "On hand " + STR(Qty,4)
		@ 7,5  SAY "On order " + STR(On_order,4)
		@ 8,5  SAY "Reorder   " + STR(Reorder,3)
		@ 9,5  SAY "Unit Cost" + STR(Cost,9,2)

		* Ask user how many to order
		@ 12,5 SAY "Order how many?" ;
		    GET New_order PICT "999"
		REPLACE Order_date WITH T_Date
		READ
	ENDIF
	SKIP
ENDDO(Continue loop untill end of file)
```

ويلاحظ من هذه الأوامر أن الكمية الجديدة التى يتم طلبها تخزن فى الحقل (New_order). كما أن حقل تاريخ الطلب (Order_date) يتم تغييره بتاريخ اليوم الحالى الذى يتم فيه طلب الأصناف. وهو التاريخ الذى يتم إدخاله عند

تشغيل نظام المخازن من البداية.

والجزء الثالث من البرنامج يتم من خلاله نسخ جميع بيانات الأصناف المطلوبة فى ملف مؤقت (Temporary File) يسمى (temp).

ويتم ذلك من خلال السطور التالية :

```
CLEAR
? "Preparing Files ..... please wait"
?
? "(Prepare printer while waiting)"
COPY TO Temp FOR New_order > 0
```

والجزء الرابع يتم من خلاله تعديل بيانات الملف الرئيسى (Master.dbf) بناء على الكميات الجديدة التى تم طلبها من بعض الأصناف. حيث يتم تعديل بيانات حقل الكمية تحت الطلب (On_order) بإضافة الكمية الجديدة التى تم طلبها إلى الكمية السابقة. ثم يتم إعادة حقل الكمية الجديدة (New_order) إلى الصفر. ويتم ذلك من خلال السطور التالية :

```
* - -Update On_order field in Master file with new
* - -orders,then set the New_order field back to zero.
REPLACE ALL On_order WITH On_order + New_order
REPLACE ALL New_order WITH 0
```

والجزء الخامس يتم من خلاله فتح الملف المؤقت (Temp) كما يتم فهرسته على حقل البائع (Vendor) وذلك حتى يمكن طباعة طلب الشراء الخاص بكل بائع على حدة. ويتم ذلك من خلال السطور التالية :

```
USE Temp
INDEX ON UPPER(Vendor) TO Temp
```

والجزء السادس يتم من خلاله توجيه المستخدم لتجهيز الطابعة ثم تشغيل الطابعة وإنشاء حلقة تكرارية لطباعة أوامر الشراء لكل بائع (Vendor) على

حدة. كما يتم إنشاء حلقة تكرارية أخرى داخلها لطباعة الأصناف الخاصة بكل بائع فى طلب الشراء الخاص به. ويتم ذلك من خلال السطور التالية :

```
CLEAR
? CHR(7)
WAIT "Ready printer and press any key to print orders"
SET PRINT ON
GO TOP
* - - - - - Loop through Temp file
DO WHILE .NOT. EOF()
```

ومن خلال الحلقة التكرارية الأولى يتم تخزين إسم البائع فى متغير الذاكرة (This_loop). كما يتم إنشاء متغير الذاكرة (Mtotal) لحساب المجموع الكلى لأسعار الأصناف فى كل طلب شراء كما يتم طباعة إسم البائع وعنوانه فى بداية طلب الشراء. والسطور التالية توضح ذلك :

```
This_loop = Vendor
Mtotal = 0
? vendor
? Vendor_add
?
?
? "Please send us the following items...."
```

ومن خلال الحلقة التكرارية الثانية يتم عرض بيانات الأصناف الخاصة بكل بائع (Vendor) على حدة وذلك بالنسبة لكل بائع يتم تخزين إسمه فى المتغير (This_loop). والسطور التالية توضح هذه الحلقة :

```
DO WHILE Vendor = this_loop .AND. .NOT. EOF()
	? New_order ,P_name , Cost , New_order * Cost
	Mtotal = New_order * Cost + Mtotal
	SKIP
ENDDO
```

وعند الإنتهاء من بائع معين أى قبل الإنتقال إلى إسم بائع جديد يتم طباعة أمر الشراء. كما يتم طباعة التكلفة الكلية لهذا الطلب والعنوان الذى يتم إرسال الأصناف إليه. ثم يتم نقل ورقة الطباعة (Eject).

والسطور التالية توضح هذه العملية :

```
? "Total cost :        ", Mtotal
?
? "Mail to : My company"
? "       12 - Tayaran street"
EJECT
```

ثم تستمر الحلقة التكرارية الخارجية حتى يتم طباعة باقى أوامر الشراء. وبعد ذلك يتم إعادة الطابعة إلى وضعها الأصلى (Off) والعودة إلى البرنامج الرئيسى (Master.dbf) وذلك من خلال السطور التالية :

```
ENDDO(Continue for each vendor in Temp file)
SET PRINT OFF
RETURN
```

١٤ - ٥ برنامج تعديل الملف الرئيسى

يتم الإنتقال إلى برنامج تعديل الملف الرئيسى عن طريق اختيار الرقم (3) فى قائمة برنامج تشغيل الملف الرئيسى حيث يتم تنفيذ البرنامج (MEdit.prg). والخطوات الأولية (PSEUDOCODE) لهذا البرنامج تكون كالآتى :

١ - يتم فتح الملف الرئيسى (Master.dbf) وملف الفهرس الخاص به.
٢ - يتم تكوين حلقة تكرارية لإجراء التعديل.
٣ - البحث عن رقم الجزء (Part number) المطلوب تعديل بياناته.
٤ - عند عدم إدخال رقم الجزء (Part number) يتم الرجوع إلى القائمة.
٥ - عند العثور على الجزء المطلوب يتم عرض بياناته باستخدام شاشة الإدخال (IScreen1).

٦ - إذا لم يتم العثور على الجزء المطلوب يتم تحذير المستخدم وإعطاؤه الفرصة لإدخال رقم جديد.

٧ - الإستمرار فى التعديل حتى يطلب المستخدم الخروج.

٨ - عند الإنتهاء يتم الرجوع إلى القائمة الرئيسية لتشغيل الملف الرئيسى.

والبرنامج الذى يحقق هذه الخطوات الأولية يتم كتابته كالآتى :

```
******************************** MEdit.prg
*   Edit the inventory Master file
*   Called from Master menu , MMenu.prg .
USE Master INDEX Master

*----- Start loop for performing edits
Partnumb = "X"
DO WHILE Partnumb # " "

    *--- Find out what part number to edit.
    CLEAR
    @ 2,1  SAY "Edit Inventory Master File"
    @ 2,60 SAY DTOCT(T_Date) + "   " + TIME()
    @ 3,0  SAY Uline
    ?
    ?
    ?
    ?
    Partnumb = SPACE(5)
    @ 15,5 SAY "Edit for what part number?" ;
        GET Partnumb
    READ

    *----- Try to find that part number
    Partnumb = UPPER(Partnumb)
    SEEK Partnumb
```

```
    DO CASE

        * - - - If no part number entered, return to
        * - - - Master menu.
        CASE Partnumb = " "
            CLEAR

        * - - - If part number found , edit using
        * - - - Iscreen1 format file
        CASE FOUND()
            SET FORMAT TO Iscreen1
            READ
            SET FORMAT TO

        * - - - Otherwise warn user, and allow another try
        CASE .NOT. FOUND()
            @ 17,5 SAY "There is no part" + "Partnumb"
            @ 24,5 SAY "Press any key to try again"
            WAIT
    ENDCASE
ENDDO(Continue editing untill user requests ...exit)
* - - - - - Return to Master menu
RETURN
```

والجزء الأول من البرنامج يبدأ كالعادة بالتعريف بإسم البرنامج ووظيفته ثم إسم الملف الذى يقوم باستدعائه. ثم يتم فتح الملف الرئيسى وملف الفهرس الخاص به. وكالعادة يتم تكوين حلقة تكرارية للبحث عن رقم الجزء (Part number) الذى يدخله المستخدم فى المتغير (Partnumb) ويتم ذلك من خلال السطور التالية :

```
USE Master INDEX Master
Partnumb = "X"
DO WHILE Partnumb # " "
```

ثم يتم عرض عنوان الشاشة والتاريخ والوقت ثم سؤال المستخدم عن الرقم المطلوب. وذلك من خلال السطور التالية :

```
CLEAR
@ 2,1  SAY "Edit Inventory Master File"
@ 2,60 SAY DTOCT(T_Date) + "   " + TIME()
@ 3,0  SAY Uline
?
?
Partnumb = SPACE(5)
@ 15,5 SAY "Edit for what part number?" GET Partnumb
READ
```

ثم يتم تحويل رقم الجزء إلى حروف كبيرة (Uppercase) والبحث عنه بواسطة الأمر (SEEK). وذلك كالآتى :

```
Partnumb = UPPER(Partnumb)
SEEK Partnumb
```

ثم يبدأ البرنامج بعد ذلك فى اتخاذ القرار بناء على رقم الجزء الذى يدخله المستخدم. فإذا لم يدخل المستخدم أى رقم يتم مسح الشاشة والخروج من الحلقة التكرارية مما يؤدى إلى الرجوع إلى القائمة الرئيسية لتشغيل الملف الرئيسى. أما إذا أدخل رقما وتم العثور على هذا الرقم (FOUND()) فإن البرنامج يعرض بيانات هذا الجزء من خلال شاشة الإدخال (IScreen1) التى سبق إنشاؤها ثم يتيح للمستخدم تعديل بيانات هذا الجزء. ثم يتم إغلاق ملف شاشة الإدخال (IScreen1).

وإذا أدخل المستخدم رقما ولكن لم يتم العثور على هذا الرقم فإن البرنامج يحذر المستخدم ويعطيه الفرصة لإدخال رقم آخر.

والسطور التالية توضح هذه العملية :

```
DO CASE
   CASE Partnumb = " "
      CLEAR
  CASE FOUND()
      SET FORMAT TO Iscreen1
      READ
      SET FORMAT TO
  CASE .NOT. FORUND()
      @ 17,5 SAY "There is no part " + Partnumb
      @ 24,5 SAY "Press any key to try again"
      WAIT
ENDCASE
ENDDO(Continue editing untill user requests exit)
RETURN
```

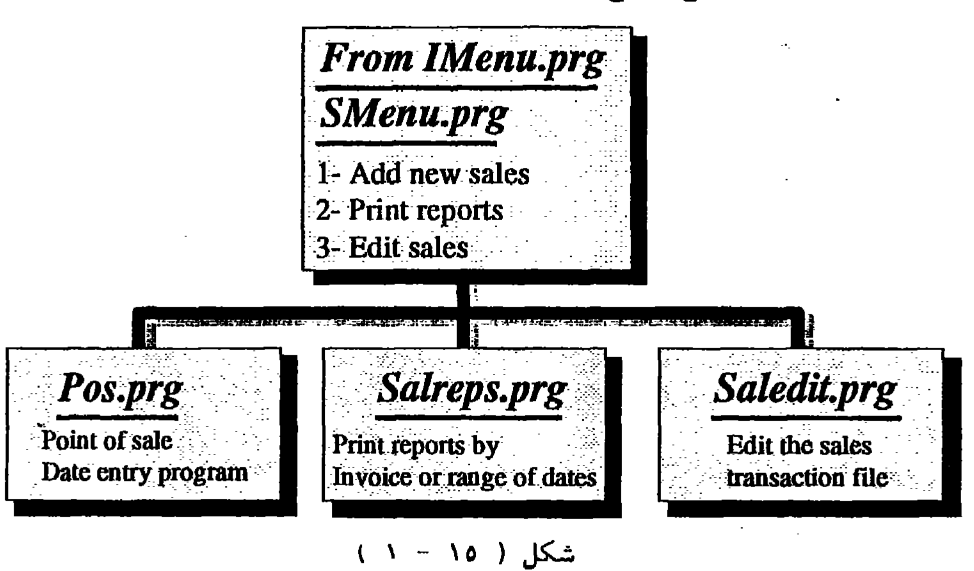
الفصل الخامس عشر

برنامج تشغيل ملف المبيعات

هذا الجزء من برنامج المخازن يختص بمتابعة مبيعات الأصناف وتحديث ملف المبيعات بناء على ذلك. ويتم ذلك عن طريق برنامج مكون من أربعة برامج منفصلة كما سيتم الإيضاح فيما بعد. ويقوم هذا البرنامج بالإشراف على نقطة البيع (Point of Sale) حيث يقوم بتجهيز فواتير البيع بناء على رقم الجزء الذى يدخله المستخدم. ويقوم البرنامج كذلك بملء بيانات الأصناف فى الفاتورة آليا. كما يقوم البرنامج بتحذير المستخدم إذا أدخل رقم جزء غير موجود ويتيح له إدخال رقم جديد. كما يقوم البرنامج بعد ذلك بتجميع أسعار الأجزاء فى الفاتورة بالإضافة إلى تخزين أرقام الفواتير.

١٥ - ١ تركيب برنامج المبيعات

يتكون هذا البرنامج من أربعة برامج كما سبق الإيضاح. وهى برنامج(SMenu.prg) الذى يتحكم فى البرامج الثلاث الأخرى وبرنامج (Pos.prg) الذى يتحكم فى نقطة البيع (Point of sale) وبرنامج (Salreps.prg) الذى يطبع التقارير التى توضح موقف مبيعات الأصناف وبرنامج (Saledit.prg) الذى يسمح بتعديل بيانات ملف المبيعات. والتركيب الهرمى للبرنامج يتضح من الشكل التالى :



شكل (١٥ - ١)

وفى الجزء التالى سوف يتم شرح برامج (SMenu) و (Pos) و (Salreps) مع تأجيل شرح برنامج (Saledit) إلى الفصل الخاص بتحديث الملفات.

١٥ - ٢ برنامج القائمة الرئيسية (Smenu.prg)

يتم تشغيل هذا البرنامج عندما يختار المستخدم الاختيار رقم (2) فى القائمة الرئيسية لنظام المخازن. وعند تشغيل البرنامج تظهر القائمة التالية على الشاشة.

```
Sales System Menu      02/20/90      04:50:30

   1 .   Enter point of sale routine
   2 .   Print sales reports
   3 .   Edit sales data
   4 .   Return to main menu

         Enter choice (1 - 4 )
```

شكل (١٥ - ٢)

والبرنامج لايختلف تركيبه عن أى برنامج قائمة من البرامج السابقة. ولذلك فليست هناك حاجة إلى كتابة الخطوات الأولية (PSEUDOCODE) له ، ونكتفى فقط بعرض سطور البرنامج كالآتى :

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * SMenu.prg
*   Menu of sales portion of the inventroy system
*   Called from inventory system Main menu .

* - - - - - Set up loop for presenting menu
Schoice = 0
DO WHILE Schoice # 4
   CLEAR
   @ 2,1  SAY "Sales system menu"
   @ 2,60 SAY DTOC(T_Date) + "  " + TIME()
```

```
@ 3,0  SAY Uline
?
?
TEXT

        1. Enter point of sale routine
        2. Print sales reports
        3. Edit sales data
        4. Return to main menu

ENDTEXT
@ 24,1 SAY "Enter choice (1 - 4) " GET Schoice PICT "9"
READ

* - - - - -  Branch to appropriate choices
DO CASE
     CASE Schoice = 1
            DO Pos
     CASE Schoice = 2
            DO Salreps
     CASE Schoice = 3
            DO Saledit
     ENDCASE
ENDDO(While schoice #4)
* - - - - -  when done , return to main menu
RETURN
```

١٥ - ٣ برنامج نقطة البيع (Pos.prg)

قبل البدء فى شرح أوامر هذا البرنامج سيتم أولا شرح ما يظهر على الشاشة عند تنفيذه حتى يكون القارىء متتبعا لوظائف هذا البرنامج. فعند تشغيل البرنامج تظهر شاشة تشبه فاتورة البيع تماما كما يتضح من الشكل (١٥ - ٣).

02/20/90	Invoice number :
Clerk ▒▒▒▒	Customer : ▒▒▒▒

Part #	Name	Qty	Price	Total

شكل (١٥ - ٣)

والبرنامج يملأ البيانات الموجودة أعلى الفاتورة آليا مثل التاريخ ورقم الفاتورة كما يقوم المستخدم بكتابة إسم الموظف القائم بعملية البيع(Clerk) وإسم العميل الذى يتم البيع له. ثم يظهر عمود ضوئى لإدخال رقم الجزء فيه كما يتضح من الشكل (١٥ - ٤).

02/20/90	Invoice number :
Clerk ▒▒▒▒	Customer : ▒▒▒▒

Part #	Name	Qty	Price	Total
▒▒▒▒				

شكل (١٥ - ٤)

وعند إدخال المستخدم لرقم جزء غير موجود يظهر التحذير التالى :

No such part !!

ويمكن للمستخدم فى هذه الحالة أن يحاول مرة ثانية. وعند كتابة رقم جزء موجود فإن إسم هذا الجزء يظهر آليا كما تظهر أعمدة ضوئية (Highlights) لإدخال الكمية (Qty) وسعر البيع (Selling Price). كما يقوم البرنامج بحساب السعر الكلى عن طريق ضرب الكمية (Qty) فى سعر البيع (Price) ثم يظهر العمود الضوئى الخاص برقم الجزء التالى حتى يقوم المستخدم بإدخال رقم جديد. وتتضح هذه العملية من الشكلين التاليين

02/20/90 *Invoice number :*

Clerk *Customer :*

Part #	Name	Qty	Price	Total
A-121	Shoe	30	20	

شكل (١٥ - ٥)

02/20/90 *Invoice number :*

Clerk *Customer :*

Part #	Name	Qty	Price	Total
A-121	Shoe	30	20	600

شكل (١٥ - ٦)

وعند إدخال المستخدم لرقم جزء جديد موجود فى المخزن بالإضافة إلى إدخال الكمية والسعر لهذا الصنف يقوم المستخدم بتنفيذ نفس العملية السابقة ويتضح ذلك من الشكل

التالى :

02/20/90		Invoice number :		
Clerk		Customer :		
Part #	Name	Qty	Price	Total
A-121	Shoe	30	20	600
A-122	Shirt	20	25	500

شكل (١٥ - ٧)

وعندما يدخل المستخدم رقما موجودا ولكنه ليس الجزء الذى يريده (وهذا يتضح له من إسم الجزء الذى يظهر آليا) فيمكنه فى هذه الحالة الضغط على مفتاح (<--) مرتين للرجوع إلى مكان رقم الجزء، وكتابة رقم جزء آخر. كما يمكن الضغط على مفتاح الإدخال مرتين لتنفيذ نفس العملية السابقة. حيث أن الضغط على مفتاح الإدخال دون إدخال أى عدد فى الكمية (Qty) أو فى السعر (Price) يخبر البرنامج أن هذا الجزء غير مطلوب. وبالتالى يعود المؤشر إلى مكان رقم الجزء لإدخال رقم جزء جديد.

وعندما ينتهى المستخدم من إدخال بيانات هذه الفاتورة (Invoice) فإنه يضغط على مفتاح الإدخال عندما يكون المؤشر واقفا على العمود الضوئى الخاص برقم الجزء(Part #). وفى هذه الحالة يقوم البرنامج بتجميع السعر الكلى للفاتورة كما يعرض رسالة للمستخدم لسؤاله إذا كان يريد طباعة هذه الفاتورة أم لا. ويتضح هذا من الشكل (١٥ - ٨)

وعندما يكتب المستخدم (Y) يتم طباعة هذه الفاتورة (Invoice) ثم يقوم البرنامج بمسح الشاشة وعرض سؤال للمستخدم إذا كان يريد طباعة فاتورة أخرى فإذا أجاب بنعم (Y) يتم عرض فاتورة جديدة بنفس الطريقة وبرقم فاتورة جديد. وإذا أجاب بلا (N) يتم الرجوع إلى قائمة البيع.

02/20/90		Invoice number :		
Clerk		Customer :		
Part #	Name	Qty	Price	Total
A-121	Shoe	30	20	600
A-122	Shirt	20	25	500
				Total : 1100
Print invoice ? (Y/N)				

شكل (١٥ - ٨)

١٥ - ٣ - ١ كتابة الخطوات الأولية (PSEUDOCODE)

كما سبق الإيضاح فإن البرنامج الخاص بالتحكم فى نقطة البيع (Point of sale) يتم تسميته (Pos.prg). ويتم كتابة الخطوات الأولية له كالآتى :

١ - يتم الحصول على آخر رقم فاتورة من ملف المبيعات (Sales.dbf).
٢ - يتم إنشاء ملف مؤقت (Temporary File) لمتابعة حركة البيع.
٣ - يتم فتح الملف الرئيسى والملف المؤقت.
٤ - يتم ربط الملف الرئيسى بالملف المؤقت.
٥ - يتم تكوين حلقة تكرارية لعرض الفواتير على الشاشة حيث يتم أولا عرض السطور الأولى من التقرير.
٦ - يتم تكوين حلقة تكرارية لكل جزء يتم إدخاله إلى الفاتورة.
٧ - يتم تكوين حلقة تكرارية لاختبار رقم الجزء والتأكد من وجوده فى الملف الرئيسى (Master.dbf).
٨ - عند عدم إدخال أى رقم يتم الخروج من البرنامج.

٩ - عند إدخال رقم غير موجود يتم تحذير المستخدم لإدخال رقم جديد.
١٠- يتم إدخال الكمية والسعر لكل جزء.
١١- عند إدخال كمية (صفر) يتم إدخال رقم جديد.
١٢- يتم حساب السعر الكلى للصنف وعرضه تحت العنوان (Total).
١٣- يتم تخزين هذه الحركة فى ملف مؤقت (Temporary File).
١٤- يتم زيادة عداد رقم السطر لعرض بيانات الصنف التالى.
١٥- عند الوصول إلى آخر سطر فى الشاشة يتم زحزحة الشاشة (Scrolling) سطرا واحدا إلى أعلى.
١٦- يتم الإستمرار فى تنفيذ الحلقة التكرارية حتى يتوقف المستخدم عن إدخال الأصناف. وفى هذه الحالة يتم عرض السعر الكلى للفاتورة (Grand Total) ثم يتم عرض سؤال للمستخدم عما إذا كان يريد طباعة الفاتورة أم لا.
١٧- يتم سؤال المستخدم إذا كان يريد طباعة فاتورة أخرى وبناء على ذلك تستمر الحلقة التكرارية حتى يطلب المستخدم الخروج.
١٨- يتم إغلاق ملفات قواعد البيانات (Database Files) كما يتم تحديث ملف المبيعات (Sales File).
١٩- يتم مسح الملف المؤقت (Temporary File).
٢٠- يتم الرجوع إلى القائمة الرئيسية للمبيعات.

١٥ - ٣ - ٢ كتابة البرنامج

قبل كتابة البرنامج يجب ملاحظة أن هذا البرنامج طويل بعض الشىء. وربما يحتاج إلى برنامج معالجة كلمات آخر غير البرنامج الموجود مع برنامج (+ DBase III) وذلك فى حالة كتابته متضمنا كل سطور الملاحظات. ولذلك يفضل عدم كتابة سطور الملاحظات خلال البرنامج إذا تم استخدام المصحح الخطى الخاص ببرنامج (+DBase III). ولكن سنضيف سطور الملاحظات هنا للتوضيح فقط.

ملاحظة

عند الإنتهاء من جميع البرامج الخاصة بنظام المخازن يمكن مسح جميع سطور الملاحظات وذلك بعد نسخ البرامج الأصلية فى قرص آخر. حيث أن ذلك

يؤدى إلى سرعة تنفيذ البرنامج بدرجة كبيرة.

ويتم كتابة سطور البرنامج كالآتى :

```
* * * * * * * * * * * * * * Pos.prg
*   Point of sale data entry program for sales
*   Called from Sales menu , SMenu.prg

* - Get last-used invoice number from the Sales file.
USE Sales
GO BOTT
Minvoice = Invoice_no

* - - - - -  Add new transactions to a temporary file
SET SAFETY OFF
COPY STRUCTURE TO Tempinv

* - - - - -  Open Master and temporary files
SELECT A
USE Master INDEX Master
SELECT B
USE Tempinv
SET RELATION TO part_no INTO Master
* - - - - -  Set pointer in temporary file
StartTrans = 1

* - - - - -  Set up loop for displaying invoice forms
Again = "Y"
DO WHILE Again = "Y"
        * - - Set up top portion of invoice on the screen
        CLEAR
        Minvoice = Minvoice + 1
        STORE SPACE (30) TO MClerk, MCust
```

```
Mtotal = 0
@ 1,2  SAY T_Date
@ 1,30 SAY "Invoice number:" + STR(Minvoice,5)
@ 2,2  SAY "Clerk" GET MClerk
@ 2,35 SAY "Customer:" GET MCust
@ 3,0  SAY Uline
? "PART #    Name", SPACE(20)
?? "Qty      Price      Total"
READ
* - - - Set up loop for each item in the invoice
Row = 7
Adding = .T.
DO WHILE Adding
        Partnumb = SPACE(5)
        ok = .F.
        * - - - loop to check validity of part number
         DO WHILE .NOT. ok
                * - - - Set up invoice memory variables.
                Quantity = 0.00
                Sel_Price = 0.00
                * - - - - -  Ask for part number.
                Partnumb = SPACE(5)
                @ Row,2 GET Partnumb
                READ
                * - - - - -  Make sure Part number exixts.
                partnumb = UPPER(TRIM(Partnumb))
                SELECT A
                SEEK Partnumb

                * - - - - -  Decide next step based on
                * - - - - -  existance of part number.
                DO CASE
```

```
* - - - - -  No part number entered
CASE LEN(partnumb) = 0
            ok = .T.
            Adding = .F.
* - - - - Part number does not exist.
CASE .NOT. FOUND()
      @ Row, 10 SAY "No such part !!"
      oK = .F.
* - - - - -  Part number exists
CASE FOUND()
      * - - - - -Display Part name, get
      * - - - - -  quantity and price
      @ Row,10 SAY P_name
      @ Row,35 GET Quantity PICT ;
      "999.99"
      @ Row,40 GET Sel_Price PICT ;
      "999.99"
      READ
      * - - - If quantity is zero , loop
      * - - - Else , Display total.
      IF Quantity = 0
                  LOOP
      ESLE
            @ Row,50 SAY Quantity * ;
         Sel_Price PICT "##,###.##"
            Mtotal = Mtotal + Quantity * ;
            Sel_price
            ok = .T.
      ENDIF

      * - - - Add a blank record to the
      * - - - Tempinv file , and fill in
```

```
                    * - - - the fields.
                    SELECT B
                    APPEND BLANK
                    REPLACE Date WITH T_Date
                    REPLACE Clerk WITH M_Clerk
                    REPLACE Invoice_no WITH Minvoice
                    REPLACE Customer WITH MCust
                    REPLACE Part_no WITH Partnumb
                    REPLACE Qty WITH Quantity
                    REPLACE Price WITH Sel_Price
                    REPLACE Posted WITH .F.
            ENDCASE
        ENDDO(continue loop for checking part number)
        Row = Row + 1
        * - - - Scroll screen if reached end of screen.
        IF Row >= 19
            @ 24,1
            ?
            Row = 19
        ENDIF
ENDDO(while still adding items to invoice)
* - - - Display grand total,and pause before next
* - - - - -  invoice.
@ Row+2,40 SAY "Total:"
@ Row+2,50 SAY Mtotal PICT "##,###.##"
Pinvoice = "Y"
@ 23,2 SAY "Print invoice ? (Y/N)" GET Pinvoice PICT "!"
READ
* - - - - -  Print invoice , Reset Start Trans.
IF Pinvoice = "Y"
        SET PRINT ON
        ? "Date:" , T_Date
```

```
            ? "Invoice number:" , STR(Minvoice,5)
            ? "Customer:" , MCust , SAPACE(20)
            ?? "Clerk:" , Mclerk
            ? Uline
            ?
            SELECT B
            GOTO StartTrans
            LIST OFF WHILE .NOT. EOF() Part_no, ;
            A --> P_name, Qty , Price, Qty * Price
            ?
            ?
            ? "Total:" , SPACE(34), Mtotal
            EJECT
            SET PRINT OFF
            StartTrans = RECCOUNT() + 1
      ENDIF
      CLEAR
      @ 23,2 SAY "Do another transaction? (Y/N)" ;
         GET Again PICT "!"
      READ
ENDDO(add invoices while user does not request exit)

* - - - - - Close databases and update sales file.
CLOSE DATABASES
CLEAR
? "Updating transaction file , please wait ..."
SET TALK ON
USE SALES
APPEND FROM Tempinv
USE Tempinv
ZAP
USE Sales INDEX Sales
```

```
REINDEX
SET TALK OFF
CLOSE DATABASES
RETURN
```

والجزء الأول من البرنامج يبدأ كالعادة بالتعريف بإسم البرنامج ووظيفته ثم إسم البرنامج الذى قام باستدعائه. ثم يتم فتح ملف المبيعات (Sales.dbf) دون استخدام أى فهرس معه. وذلك للحصول على آخر رقم فاتورة (Invoice_no) حيث أن رقم الفاتورة يتوقف على ترتيب إدخال هذه الفاتورة فى الملف. ثم يتم تخزين هذا الرقم فى المتغير (Minvoice). ويتم ذلك من خلال السطور التالية :

```
USE Sales
GO BOTT
Minvoice = Invoice_no
```

والجزء الثانى يوضح استخدام الملف المؤقت (Tempinv.dbf) فى تخزين الفواتير الجديدة قبل نقلها إلى ملف المبيعات (Sales.dbf) وتوفر هذه الطريقة سرعة كبيرة لتنفيذ البرنامج. وبعد الإنتهاء من إدخال الفواتير المطلوبة يقوم البرنامج بإضافة هذه الفواتير إلى ملف المبيعات. ويلاحظ هنا استخدام الأمر (SET SAFETY OFF) حتى يتم نسخ هيكل ملف المبيعات فى الملف المؤقت دون ظهور رسالة التحذير المعتادة فى هذه الحالة والسطور التالية توضح هذه العملية.

```
SET SAFETY OFF
COPY STRUCTURE TO Tempinv
```

والجزء الثالث يتم من خلاله فتح الملف المؤقت والملف الرئيسى فى مناطق عمل مختلفة. ثم يتم ربط الملفين بناء على حقل رقم الجزء (Part number). وهذا يساعد على اختبار رقم الجزء الذى يدخله المستخدم والتأكد من وجوده فى الملف الرئيسى. كما أن الربط بين الملفين يساعد بعد ذلك على الحصول على إسم الجزء حتى يتم كتابته فى الفاتورة. والسطور التالية توضح فتح هذين الملفين

وإنشاء العلاقة (Relation) بينهما.

```
SELECT A
USE Master INDEX Master
SELECT B
USE Tempinv
SET RELATION TO part_no INTO Master
```

وعند طباعة الفواتير يجب أن يعرف البرنامج رقم السجل الذى يبدأ بطباعته. لذلك يتم إنشاء متغير الذاكرة (StartTrans) وإعطاؤه القيمة (1) من خلال السطر التالى.

```
StartTrans = 1
```

والجزء الرابع يتم من خلاله تكوين حلقة تكرارية لإضافة الفواتير (Invoices) ويتم استخدام المتغير (Again) فى التحكم فى هذه الحلقة كما يتم من خلال الحلقة زيادة قيمة المتغير (Minvoice) بواحد عند الإنتقال إلى فاتورة جديدة.

كما يتم إنشاء المتغير (MClerk) والمتغير (MCust) لكل فاتورة جديدة لتسجيل إسم الموظف القائم بالبيع وإسم العميل فى كل فاتورة. ثم يتم عرض السطور العلوية للتقرير وذلك من خلال السطور التالية :

```
* - - - - - Set up loop for displaying invoice forms
Again = "Y"
DO WHILE Again = "Y"
   * - - Set up top portion of invoice on the screen
   CLEAR
   Minvoice = Minvoice + 1
   STORE SPACE (30) TO Mclerk, MCust
   Mtotal = 0
   @ 1,2  SAY T_Date
   @ 1,30 SAY "Invoice number:" + STR(Minvoice,5)
```

```
@ 2,2  SAY "Clerk" GET MClerk
@ 2,35 SAY "Customer:" GET MCust
@ 3,0  SAY Uline
? "PART #    name", SPACE(20)
?? "Qty      Price      Total"
READ
```

والجزء الخامس يتم من خلاله تكوين حلقة تكرارية لإدخال الأصناف فى الفاتورة. ويتم إنشاء المتغير (Adding) للتحكم فى هذه الحلقة. ويبدأ عرض الأصناف إبتداء من السطر رقم (7) على الشاشة. وتستمر الحلقة التكرارية حتى يضغط المستخدم على مفتاح الإدخال دون إدخال رقم جزء (Part number). والسطور التالية توضح هذه العملية.

```
Row = 7
Adding = .T.
DO WHILE Adding
```

والجزء السادس يتم من خلاله تكوين حلقة تكرارية أخرى لاختبار رقم الجزء الذى يدخله المستخدم وذلك بعد إنشاء متغير الذاكرة (Partnumb) وكذلك إنشاء المتغير المنطقى (ok) للتحكم فى الحلقة التكرارية. ويتم إعطاء هذا المتغير القيمة (.F.) حتى يتم تنفيذ الحلقة التكرارية مرة واحدة على الأقل. والسطور التالية توضح هذه العملية.

```
Partnumb = SPACE(5)
ok = .F.
DO WHILE .NOT. ok
```

كما يتم إنشاء متغير ذاكرة (Quantity) لتخزين الكمية المباعة من الصنف والمتغير (Sel_Price) لتخزين سعر هذا الصنف. وذلك من خلال السطور التالية :

```
Quantity = 0.00
Sel_Price = 0.00
```

واستخدام نقطة العلامة العشرية هنا مهم لتخزين الأعداد متضمنة رقمين عشريين.

والجزء السابع يتم من خلاله سؤال المستخدم عن رقم الجزء المطلوب إدخاله. ويتم تخزين هذا الرقم فى المتغير (Partnumb) ثم يتم تحويل هذا الرقم إلى حروف كبيرة (Uppercase) حتى يتم البحث عنه بواسطة الأمر (SEEK). والسطور التالية توضح هذه العملية.

```
Partnumb = SPACE(5)
@ Row,2 GET Partnumb
READ
Partnumb = UPPER(TRIM(Partnumb))
SELECT A
SEEK Partnumb
```

والجزء الثامن يتم من خلاله اتخاذ القرار بناء على ما يدخله المستخدم فى المتغير (Partnumb). فعندما يضغط المستخدم على مفتاح الإدخال دون إدخال أى رقم جزء يتم تخزين القيمة (.T.) فى المتغير (ok) والقيمة (.F.) فى المتغير (Adding) وذلك للخروج من الحلقة التكرارية حيث أن ذلك معناه أن المستخدم يريد الخروج. والسطور التالية توضح هذه العملية.

```
DO CASE
   CASE LEN(partnumb) = 0
      ok = .T.
      Adding = .F.
```

وعندما يتم إدخال رقم جزء غير موجود فى الملف يتم تحذير المستخدم وإعطاؤه الفرصة للمحاولة مرة ثانية. ويتم ذلك عن طريق تخزين القيمة (.F.) فى المتغير (ok) حتى يتم تنفيذ الحلقة التكرارية مرة ثانية. ويتم ذلك من خلال السطور التالية :

```
CASE .NOT. FOUND()
@ Row, 10 SAY "No such part !!"
oK = .F.
```

وعندما يتم إدخال رقم موجود فى الملف فإن البرنامج يأتى بباقى البيانات الخاصة بهذا الصنف مثل إسم الصنف (P_name) ثم يسأل عن الكمية المطلوبة من هذا الصنف وسعر البيع. ويتم ذلك من خلال السطور التالية :

```
CASE FOUND()
   * - - - - - Display Part name, get
   * - - - - - quantity and price
   @ Row,10 SAY P_name
   @ Row,35 GET Quantity PICT "999.99"
   @ Row,40 GET Sel_Price PICT "999.99"
   READ
```

والجزء التاسع يتم من خلاله حساب السعر لكل صنف ثم حساب السعر الكلى للفاتورة. كما يتم من خلال هذا الجزء أيضا إعادة تنفيذ الحلقة التكرارية باستخدام الأمر (LOOP) فى حالة إدخال المستخدم للكمية (صفر) وذلك عندما يضغط على مفتاح الإدخال عند وقوف المؤشر على العمود الضوئى (Highlight) الخاص بالكمية. ويتم ذلك من خلال السطور التالية :

```
IF Quantity = 0
   LOOP
ESLE
   @ Row,50 SAY Quantity * Sel_Price;
     PICT "##,###.##"
   Mtotal = Mtotal + Quantity * Sel_Price
   ok = .T.
ENDIF
```

والجزء العاشر يتم من خلاله نقل البيانات التى تم إدخالها فى متغيرات

الذاكرة إلى الحقول المقابلة فى الملف المؤقت (Tempinv). وذلك من خلال السطور التالية :

```
SELECT B
APPEND BLANK
REPLACE Date WITH T_Date
REPLACE Clerk WITH M_Clerk
REPLACE Invoice_no WITH Minvoice
REPLACE Customer WITH MCust
REPLACE Part_no WITH Partnumb
REPLACE Qty WITH Quantity
REPLACE Price WITH Sel_Price
REPLACE Posted WITH .F.
ENDCASE
ENDDO(continue loop for checking part number)
```

والجزء التالى يتم من خلاله زيادة عداد السطور بواحد لكتابة بيانات الصنف التالى. وعند الوصول إلى نهاية الشاشة يتم زحزحة الشاشة (Scrolling) بمقدار سطر واحد لأعلى. ويتم ذلك عن طريق تحريك المؤشر إلى آخر سطر على الشاشة دون كتابة أى شىء فيه وذلك مع تثبيت رقم السطر (Row) عند الرقم (١٩). ويتم ذلك من خلال السطور التالية :

```
Row = Row + 1
* - - - - - Scroll screen if reached end of screen.
IF Row > = 19
   @ 24,1
   ?
   Row = 19
ENDIF
ENDDO(while still adding items to invoice)
```

وعند الإنتهاء من إدخال الأصناف فى هذه الفاتورة يعرض البرنامج السعر الكلى (Total) لهذه الفاتورة ويسأل المستخدم إذا كان يريد طباعة الفاتورة أم

لا. والسطور التالية توضح هذه العملية :

```
@ Row+2,40 SAY "Total:"
@ Row+2,50 SAY Mtotal PICT "##,###.##"
Pinvoice = "Y"
@ 23,2 SAY "Print invoice ? (Y/N)" ;
  GET Pinvoice PICT "!"
READ
```

وعندما يكتب المستخدم (Y) يتم تشغيل الطابعة ثم يتم كتابة رأس التقرير (Heading) ثم يتم تحريك مؤشر السجلات (Record Pointer) إلى أول سجل فى الملف المؤقت (Tempinv) وهو السجل الذى تم تخزين رقمه فى المتغير (StartTrans). ويتم عرض بيانات الصنف فى الفاتورة وحساب السعر الكلى لهذا الصنف ثم الإنتقال إلى الصنف التالى وعرض بياناته وهكذا حتى نهاية الملف. وعند الإنتهاء من إدخال كل الأصناف فى الفاتورة يتم تقديم صفحة جديدة على الطابعة باستخدام الأمر (EJECT) للتجهيز لطباعة فاتورة جديدة عندما يريد المستخدم ذلك. ويتم تخزين الرقم المثل لعدد السجلات فى الملف المؤقت (Tempinv) زائدا واحد فى متغير الذاكرة (StartTrans). وذلك لكى تبدأ الفاتورة التالية من هذا الرقم عندما يريد المستخدم إرسال فاتورة أخرى بإسم بائع آخر (Vendor). والسطور التالية توضح هذه العملية.

```
IF Pinvoice "Y"
   SET PRINT ON
   ? "Date:" , T_Date
   ? "Invoice number:" , STR(Minvoice,5)
   ? "Customer:" , MCust , SPACE(20)
   ?? "Clerk:" , MClerk
   ? Uline
   ?
   SELECT B
   GOTO StartTrans
   LIST OFF WHILE .NOT. EOF() Part_no, A -> P_name, ;
   Qty , Price, Qty * Price
```

```
    ?
    ?
    ? "Total:" , SPACE(34), Mtotal
    EJECT
    SET PRINT OFF
    StartTrans = RECCOUNT() + 1
ENDIF
```

وفى الجزء التالى يتم سؤال المستخدم إذا كان يريد طباعة فواتير أخرى. فإذا أراد ذلك يتم تنفيذ الحلقة مرة ثانية حيث يكون المتغير (Again) مخزنا فيه القيمة ("Y"). أما إذا أراد الخروج فإنه يكتب (N) وفى هذه الحالة يتوقف تنفيذ الحلقة. ويتم ذلك من خلال السطور التالية :

```
CLEAR
@ 23,2 SAY "Do another transaction? (Y/N)" ;
  GET Again PICT "!"
READ
ENDDO(add invoices while user does not request exit)
```

وفى الجزء التالى يقوم البرنامج بإضافة السجلات الموجودة فى الملف المؤقت (Tempinv) إلى آخر ملف المبيعات كما يتم تحديث ملف الفهرس. ويقوم البرنامج أيضا بمسح السجلات الموجودة فى الملف المؤقت لتوفير المساحة التخزينية على القرص. ثم يقوم البرنامج بإغلاق جميع الملفات المفتوحة والعودة إلى البرنامج الرئيسى لتشغيل برنامج المبيعات (SMenu.prg). ويتم ذلك من خلال السطور التالية :

```
CLOSE DATABASES
CLEAR
? "Updating transaction file , please wait ..."
SET TALK ON
USE SALES
APPEND FROM Tempinv
```

```
USE Tempinv
ZAP
USE Sales INDEX Sales
REINDEX
SET TALK OFF
CLOSE DATABASES
RETURN
```

١٥ - ٣ - ٣ إدخال السعر آليا

يجدر الإشارة هنا إلى أن هذا البرنامج عام يمكن استخدامه فى أى نقطة بيع لذلك فإنه يبدو طويلا ومعقدا بعض الشىء. كما يمكن تعديله ليناسب الظروف والمطالب المختلفة للمستخدم. حيث يمكن مثلا إدخال السعر آليا فى فاتورة البيع (Invoice) بدلا من كتابته بواسطة المستخدم. ولإجراء هذا التعديل يلزم أولا إضافة حقل سعر الجزء (Price) فى الملف الرئيسى (Master.dbf) ثم تعديل البرنامج بناء على ذلك.

١٥ - ٤ برنامج تقارير البيع

يتم تنفيذ هذا البرنامج عندما يختار المستخدم الرقم (2) من قائمة تشغيل ملف المبيعات (SMenu). وهذا البرنامج يتيح للمستخدم عرض أو طباعة تقارير إما لفاتورة محددة أو لحركة المبيعات خلال فترة محددة (أى من تاريخ محدد إلى تاريخ آخر). وعند تشغيل هذا البرنامج تظهر الشاشة التالية :

```
Sales Report Options            02/20/90  09:30:25

        1.  By invoice number
        2.  By dates
        3.  Return to sales menu

    Enter  choice  ( 1 - 3 )
```

شكل (١٥ - ٩)

وعندما يختار المستخدم الرقم (١) من قائمة برنامج تقارير مبيعات الأصناف لعرض أو طباعة تقرير عن فاتورة محددة يظهر السؤال التالى :

Look for what invoice number?

وعندما يقوم المستخدم بإدخال رقم الفاتورة (Invoice number) يظهر التقرير الخاص بهذه الفاتورة على الشاشة أو على الطابعة كالآتى مثلا :

Invoice number : 1246 **Date : 02/20/90**
Clerk : Hassan **Customer : D. Fathy**

Part #	Qty	Part name	Price	Total
AAA	1	Printer	900	900
BBB	10	Floppy disk	16	160

شكل (١٥ - ١٠)

وعندما يختار المستخدم الرقم (2) من القائمة لعرض بيانات حركة البيع خلال فترة محددة يظهر الآتى على الشاشة :

Enter start date : / /
Enter end date : / /

وعندما يكتب المستخدم تاريخ البداية وتاريخ النهاية يظهر التقرير الخاص بحركة المبيعات خلال هذه الفترة على الشاشة أو على الطابعة كالموضح بالشكل (١٥ - ١١)

ويمكن إنشاء التقرير باستخدام الأمر (CREATE) أو الأمر (MODIFY) وذلك كالآتى :

CREATE REPORT Sales

على أن تكون محتويات الأعمدة كالموضح بالشكل (١٥ - ١٢)

Page No. 1 02/20/90	Sales Transactions					
Date	Invoice	Sales person	Customer	Part	Qty	Sale Price
02/01/90	1243	Magdy	Salem	SAF	20	700
02/10/90	1360	Medhat	Shawky	LAM	10	500
TOTAL				1200		

شكل (١٥ - ١١)

Column	Contents	Heading	Width	Decimals	Total
1	Date	Date	9		
2	Invoice_no	Invoice	7		
3	Clerk	Salesperson	14		
4	Customer	Customer	14		
5	Part_no	Part	6		
6	Qty	Qty	5	0	N
7	Cost	Sale Price	10	0	Y

شكل (١٥ - ١٢)

١٥ - ٤ - ١ كتابة الخطوات الأولية للبرنامج

كما سبق الإيضاح فإن الوظيفة الرئيسية لبرنامج تقارير المبيعات (Salreps.prg) هى سؤال المستخدم عن نوع التقرير الذى يريده. ثم عرض هذا التقرير على الشاشة أو طباعته على الطابعة. والخطوات الأولية لهذا البرنامج

كالآتى :

١ - يتم فتح ملف المبيعات (Sales.dbf) والملف الرئيسى (Master.dbf).
٢ - يتم ربط الملفين.
٣ - يتم تكوين حلقة تكرارية لعرض قائمة الإختيارات.
٤ - يتم سؤال المستخدم عن نوع التقرير المطلوب.
٥ - فى حالة طلب التقرير بناء على الفاتورة يتم سؤال المستخدم عن رقم الفاتورة المطلوبة.
٦ - يتم البحث عن أول سجل يحتوى على هذا الرقم فى ملف المبيعات (Sales.dbf).
٧ - عند العثور على هذا السجل يتم طباعة رأس التقرير بناء على البيانات الموجودة فى هذا السجل.
٨ - يتم طباعة بيانات الأصناف الخاصة برقم الفاتورة المطلوبة.
٩ - فى حالة طلب التقارير خلال فترة زمنية محددة يتم سؤال المستخدم عن تاريخ البداية وتاريخ النهاية.
١٠- يتم تحويل التواريخ التى يدخلها المستخدم من الصورة الحرفية إلى الصورة التاريخية.
١١- يتم طباعة التقارير خلال هذه الفترة باستخدام صورة التقرير (Sales.frm).
١٢- عند الإنتهاء يتم الرجوع إلى القائمة الرئيسية لبرنامج المبيعات (SMenu.prg).

١٥ - ٤ - ٢ كتابة البرنامج

يتكون برنامج تقارير البيع من السطور التالية :

```
* * * * * * * * * * * * * SalReps.prg
*    Print reports from the sales file
*    Called from sales menu, SMenu.prg
* - - - - -  Open files and set up relationship.
SELECT 1
USE Sales
SELECT 2
```

```
USE Master INDEX Master
SELECT 1
SET RELATION TO Part_no INTO Master
Repchoice = 0

* - - - - - Start loop for menu
DO WHILE Repchoice # 3
        CLEAR
        @ 2,1  SAY "Sales Report Options"
        @ 2,60 SAY DTOC(T_Date) + "    " + TIME()
        @ 3,0  SAY Uline
        ?
        ?
        TEXT
                1. By invoice number
                2. By dates
                3. Return to Sales menu
        ENDTEXT
        @ 24,1 SAY "Enter choice(1-3)" ;
            GET Repchoice PICT "9" RANGE 1,3
        READ

        * - - - - - IF not exiting , ask about printer.
        @ 5,0  CLEAR
        STORE "  " TO YN, Printer
        IF Repchoice # 3
                @ 15,5 SAY "Send to printer " GET YN PICT "!"
                READ
                * - - - - - Set up printer macro
                IF YN = "Y"
                        Printer = "TO PRINT"
                ENDIF
```

```
ENDIF

* - - -Print appropriate report based on request.
@ 5,0  CLEAR
DO CASE
        * - - - - Case 1 : Search by invoice number.
        CASE Repchoice = 1
                @ 15,5
                INPUT "Look for what invoice?" TO Isearch
                CLEAR
                LOCATE FOR invoice_no = Isearch
                * - - - If found , print invoice.
                IF FOUND()
                IF YN = "Y"
                        SET PRINT ON
                ENDIF
                * - -  Print header from first record
                * - - -  with that invoice number.
                ? "Invoice number : " , Invoice_no
                ?? "Date:" , Date
                ? "Clerk:", Clerk,"Customer:", Customer
                ?
                ? "Part #  Part name   Qty   Price   Total "

                * - - - Print data for all records with
                * - - - that invoice number.
                LIST OFF WHILE Invoice_no = Isearch ;
                Part_no,B-->P_name, Qty , Price , ;
                   (Qty * Price)
                IF YN = "N"
                        EJECT
                        SET PRINT OFF
```

```
            ENDIF
        ENDIF(found)

    * - - - - -  Case 2 : Search by dates
    CASE Repchoice = 2
        STORE SPACE(8) TO Start , End
        @ 15,5 SAY "Enter start date" GET Start ;
            PICT " 99/99/99"
        @ 17,5 SAY "Enter end date" ;
            GET End PICT "99/99/99"
        READ
        Start = CTOD(Start)
        End = CTOD(End)
        CLEAR
        * - - - - -  Print the report.
        REPORT FORM Sales FOR Date > = Start .AND.;
        Date < = End & Printer
ENDCASE

* - - - IF not going to printer, Pause the screen
IF YN # "Y" .AND. Repchoice # 3
    ?
    ?
    WAIT "Press any key to return to the reports menu"
ENDIF
ENDDO(while user does not request to exit)

* - - - - -  When done return to sales menu
SET RELATION TO
CLOSE DATABASES
RETURN
```

والجزء الأول من البرنامج يبدأ كالمعتاد بإسم البرنامج (Salreps.prg) ووظيفته والبرنامج القائم باستدعائه ثم أوامر فتح الملفات. ويلاحظ هنا أن ملف المبيعات (Sales.dbf) يتم فتحه بدون فتح الفهرس معه. وذلك لأن المطلوب طباعة تقارير تعتمد على رقم الفاتورة (Ivoice number) أو التاريخ (Date). وحيث أن البيانات يتم تخزينها فعليا فى ملف المبيعات مرتبة حسب رقم الفاتورة وتاريخ إدخالها لذلك فإن الملف يكون مرتبا بالترتيب المطلوب دون الحاجة إلى استخدام الفهرس. ولإدخال إسم الصنف فى الفاتورة يلزم فتح الملف الرئيسى (Master.dbf) وربطه بملف المبيعات. والسطور التالية توضح هذه الخطوات.

```
* * * * * * * * * * * * * * * SalReps.prg
*   Called from sales menu , SMenu.prg
* - - - - -  Open files and set up relationship.

SELECT 1
USE Sales
SELECT 2
USE Master INDEX Master
SELECT 1
SET RELATION TO Part_no INTO Master
```

والجزء الثانى يتم من خلاله تكوين حلقة تكرارية لعرض قائمة الإختيارات على المستخدم وسؤاله عن الإختيار المطلوب ثم تخزين هذا الإختيار فى متغير الذاكرة (Repchoice). والسطور التالية توضح هذه الخطوات :

```
Repchoice = 0
* - - - - -  Start loop for menu
DO WHILE Repchoice # 3
   CLEAR
   @ 2,1  SAY "Sales Report Options"
   @ 2,60 SAY DTOC(T_Date) + "    " + TIME()
   @ 3,0  SAY Uline
   ?
```

```
?
TEXT

    1. By invoice number
    2. By dates
    3. Return to Sales menu

ENDTEXT
@ 24,1 SAY "Enter choice(1-3)" ;
  GET Repchoice PICT "9" RANGE 1,3
READ
```

والجزء الثالث يوضح إنشاء متغير الذاكرة (Printer) وإدخال القيمة (TO PRINT) فى هذا المتغير لاستخدامه كماكرو لتشغيل الطابعة بعد ذلك وذلك فى حالة عدم اختيار المستخدم للرقم (3) من القائمة للخروج. ويتم ذلك من خلال السطور التالية :

```
* - - - - - IF not exiting , ask about printer.
@ 5,0 CLEAR
STORE "  " TO YN, Printer
IF Repchoice # 3
   @ 15,5 SAY "Send to printer " GET YN PICT "!"
   READ
   * - - - - - Set up printer macro
   IF YN = "Y"
      Printer = "TO PRINT"
   ENDIF
ENDIF
```

والجزء الرابع يوضح استخدام الأمر (DO CASE) فى طباعة التقرير المطلوب حسب اختيار المستخدم. فإذا أراد المستخدم الطباعة بناء على رقم الفاتورة فإن البرنامج يسأل عن رقم الفاتورة المطلوب كما يحدد أول سجل يحتوى على

هذا الرقم. ومن هذا السجل يقوم البرنامج بعرض عنوان التقرير (Heading) الذى يحتوى على إسم الموظف القائم بعملية البيع (Clerk) وإسم العميل المشترى (Customer) وتاريخ البيع. ثم يستخدم الأمر (LIST) والعبارة (WHILE) لعرض أو طباعة بيانات جميع السجلات التى تشترك فى رقم الفاتورة. والسطور التالية توضح هذه العملية :

```
* - - - - Print appropriate report based on request.
@ 5,0  CLEAR
DO CASE

   * - - - - -  Case 1 : Search by invoice number.
   CASE Repchoice = 1
   @ 15,5
   INPUT "Look for what invoice  ? " TO Isearch
   CLEAR
   LOCATE FOR invoice_no = Isearch
   * - - - - -  If found , print invoice.
   IF FOUND()
     IF YN = "Y"
        SET PRINT ON
     ENDIF
     * - - - - -  Print header from first record
     * - - - - -  with that invoice number.
     ? "Invoice number : " , Invoice_no
     ?? "Date:" , Date
     ? "Clerk:" , Clerk, "Customer:" , Customer
     ?
     ? "Part #  Part name    Qty   Price   Total "
     * - - -  Print data for all records with that
     * - - - - -  invoice number.
     LIST OFF WHILE Invoice_no = Isearch  ;
      Part_no,B-->P_name, Qty , Price , (Qty * Price)
```

```
    IF YN = "N"
       EJECT
       SET PRINT OFF
    ENDIF
  ENDIF(found)
```

والجزء الخامس يتم من خلاله عرض بيانات التقرير فى حالة طلب المستخدم تقريرا بالمبيعات التى تمت خلال فترة معينة. وفى هذه الحالة يسأل البرنامج عن تاريخ البداية (Start) وتاريخ النهاية (End) ثم يستخدم صورة التقرير (Sales.frm) التى سبق إنشاؤها. ويتم ذلك من خلال السطور التالية :

```
* - - - - - - - Case 2 : Search by dates
CASE Repchoice = 2
        STORE SPACE(8) TO Start , End
        @ 15,5 SAY "Enter start date" GET Start ;
        PICT " 99/99/99"
        @ 17,5 SAY "Enter end date" ;
        GET End PICT "99/99/99"
        READ
        Start = CTOD(Start)
        End = CTOD(End)
        CLEAR
        * - - - - -  Print the report.
        REPORT FORM Sales FOR Date > = Start .AND. ;
        Date < = End & Printer
ENDCASE
```

ويلاحظ هنا عملية تحويل التاريخ من الصورة الحرفية التى يدخلها المستخدم إلى الصورة التاريخية التى يستطيع البرنامج التعامل معها وذلك باستخدام الدالة (CTOD).

والجزء السادس يتم من خلاله إيقاف الشاشة لحظيا (Pause) فى حالة عدم الرغبة فى طباعة التقرير والإكتفاء بعرضه على الشاشة. ويتم ذلك من خلال السطور التالية :

```
* - - - - IF not going to printer, Pause the screen
IF YN # "Y" .AND. Repchoice # 3
   ?
   ?
   WAIT "Press any key to return to the reports ;
   menu"
ENDIF
```

والجزء السابع يتم من خلاله إنهاء الحلقة التكرارية وإغلاق الملفات المفتوحة ثم العودة إلى البرنامج القائم بالإستدعاء (SMenu.prg). ويتم ذلك من خلال السطور التالية :

```
ENDDO(while user does not request to exit)
* - - - - -  When done return to sales menu
SET RELATION TO
CLOSE DATABASES
RETURN
```

الفصل السادس عشر

برنامج تشغيل ملف الإضافة

هذا البرنامج هو جزء من برنامج المخازن (Inventory) يختص بتسجيل بيانات الأصناف التى يتم توريدها إلى المخزن. ويتم تخزين بيانات هذه الأصناف فى ملف الإضافة (Newstock.dbf) الذى سبق إنشاؤه. ويتم تشغيل هذا البرنامج عندما يختار المستخدم الاختيار رقم (3) من القائمة الرئيسية لبرنامج المخازن. انظر الشكل (١٦ - ١).

```
Inventory system main menu                02/20/90

        1. Manage master inventory
        2. Record sales
        3. Record new stock
        4. Exit
  Enter choice [0]
```

شكل (١٦ - ١)

```
New stock system menu          02/20/90    08:30:45

        1. Record new items
        2. Print new stock data
        3. Edit new stock data
        4. Exit
  Enter choice (1 - 4)
```

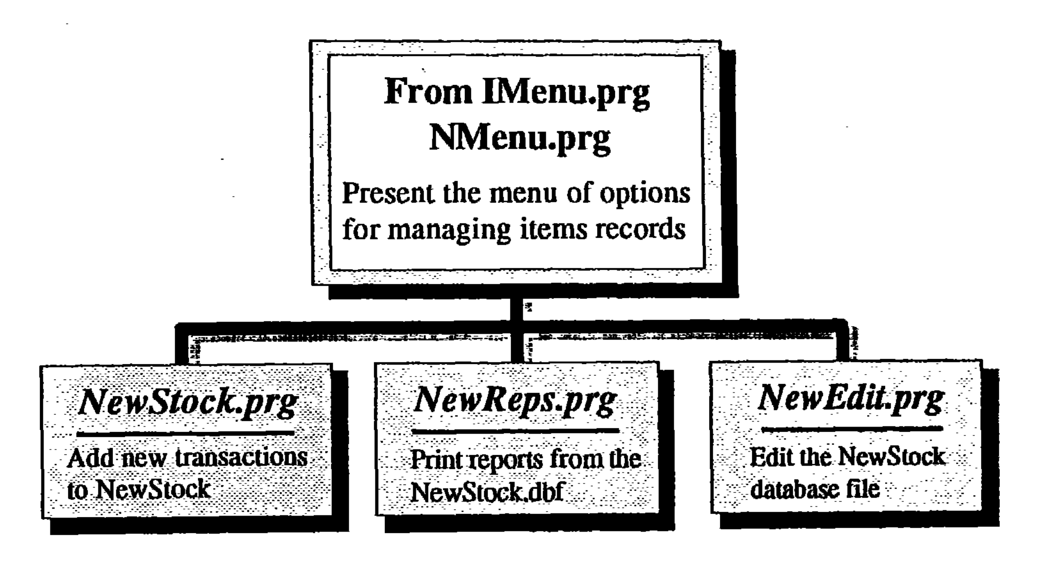
شكل (١٦ - ٢)

وفى هذه الحالة تظهر القائمة الخاصة بالإضافة. أنظر الشكل (١٦ - ٢). وهذه القائمة تتيح للمستخدم إضافة حركة جديدة (Transaction) أو طباعة تقارير أو تصحيح بيانات الحركة أو الرجوع إلى القائمة الرئيسية للمخازن.

١٦ - ١ تركيب البرنامج

يتكون برنامج الإضافة من أربعة برامج منفصلة. البرنامج الأول هو برنامج (NMenu.prg) الذى يقوم بالتحكم فى تشغيل البرامج الثلاثة الأخرى. وبناء على اختيار المستخدم يتم التفرع إلى البرنامج (NewStock.prg) الذى يسمح للمستخدم بإضافة بيانات الأصناف الجديدة. أو يتم التفرع إلى البرنامج (NewReps.prg) الذى يطبع التقارير. أو يتم التفرع إلى البرنامج (NewEdit.prg) الذى يسمح بتعديل بيانات أى حركة إضافة.

والشكل التالى يوضح التركيب الهرمى للبرنامج



شكل (١٦ - ٣)

١٦ - ٢ برنامج قائمة الإضافة (NMenu.prg)

هذا البرنامج لايختلف عن برامج القوائم التى سبق شرحها لذلك يتم عرض سطور البرنامج دون الحاجة إلى شرحها مرة ثانية وهى كالآتى :

```
* * * * * * * * * * * * * * NMenu.prg
```

```
* Menu for managing New Stock portion of Inventory
* system . Called from Inventory System main menu.
* - - - - - - Set up loop form presenting menu.
Nchoice = 0
DO WHILE NChoice # 4
    CLEAR
    @ 2,1  SAY "New Stock System menu"
    @ 2,60 SAY DTOC(T_Date) + "    " + TIME()
    @ 3,0  SAY Uline
    ?
    ?
    TEXT
          1. Record new items
          2. Print new stock reports
          3. Edit new stock data
          4. Return to main menu
   ENDTEXT
   @ 24,1 SAY "Enter choice(1-4)" GET Nchoice PICT "9" RANGE 1,4
   READ

   * - - - - - Branch to appropriate program.
   DO CASE
      CASE Nchoice = 1
         DO NewStock
      CASE Nchoice = 2
         DO NewReps
      CASE Nchoice = 3
         DO NewEdit
   ENDCASE
ENDDO(while Nchoice # 4)
* - - - - - Return to main menu
RETURN
```

١٦ - ٣ برنامج إدخال بيانات الأصناف (Newstock.prg)

عندما يختار المستخدم الرقم (1) من قائمة الإضافة فإن برنامج القائمة (NMenu.prg) ينتقل إلى البرنامج (NewStock.prg) لينفذه. وفى هذه الحالة يتم مسح الشاشة وتظهر الرسالة التالية للمستخدم :

```
Enter data for goods received
Part number :-
```

وعندما يكتب المستخدم رقما غير موجود فى قاعدة البيانات يتم تحذير المستخدم بصفارة (Beep) ثم تظهر رسالة توضح له عدم وجود هذا الجزء مع إعطائه الفرصة للمحاولة مرة ثانية. وذلك كالآتى :

```
Enter data for goods received
Part number :-          :No such part !!!
```

ويجب ملاحظة أن هذا البرنامج لايضيف أصناف بأرقام جديدة ولكنه يضيف كميات من أصناف موجودة أرقامها فى قاعدة البيانات. وهذا عكس البرنامج (Addnumbs.prg) المستخدم فى برنامج تشغيل الملف الرئيسى (Master.dbf).

وعند إدخال المستخدم لرقم جزء موجود يقوم البرنامج بعرض إسم هذا الجزء على الشاشة بالإضافة إلى تاريخ إدخال هذا الصنف (تاريخ اليوم الحالى) وإسم البائع (Vendor). كما يتيح له تعديل إسم البائع (Vendor) حسب الحاجة ثم يطلب من المستخدم كتابة الكمية الواردة من هذا الصنف وسعر الشراء. وذلك كالآتى :

```
Enter data for goods received
Part number  A-122        Shirt
Quantity : -            Price:
Date 02/20/90           Vendor : Hasan
```

وعند الإنتهاء من إدخال البيانات فى الحقول الخالية فإن البرنامج يسأل عن الصنف التالى المطلوب إضافته. وعند الإنتهاء من إدخال الأصناف يقوم المستخدم بالضغط على

مفتاح الإدخال للرجوع إلى قائمة الإضافة مرة ثانية.

١٦ - ٤ كتابة الخطوات الأولية (PSEUDOCODE)

يتم كتابة الخطوات الأولية للبرنامج كالآتى :

١ - يتم فتح الملف الرئيسى (Master.dbf) وملف الإضافة (NewStock.dbf).
٢ - يتم تكوين حلقة تكرارية لتسجيل بيانات الأصناف الجديدة.
٣ - عند إدخال رقم الجزء يقوم البرنامج بالبحث عن هذا الجزء فى الملف الرئيسى (Master.dbf).
٤ - عند عدم العثور على رقم الجزء يتم تحذير المستخدم والسماح له بإدخال رقم جديد.
٥ - عند العثور على رقم الجزء يتم عرض بياناته وسؤال المستخدم عن الكمية الواردة وسعر الشراء.
٦ - يتم الإستمرار فى إدخال الأصناف حتى يطلب المستخدم الخروج.
٧ - يتم العودة إلى قائمة الإضافة.

١٦ - ٥ كتابة البرنامج

يتم كتابة هذا البرنامج كالآتى :

```
* * * * * * * * * * * * * * * * NewStock.prg
*   Data entry program for goods received
*   Called from New Stock, NMenu.prg
SELECT A
USE Master INDEX Master
SELECT B
USE NewStock INDEX NewStock
SELECT A

* - - - - - Set up loop for recording goods received.
Partnumb = "x"
DO WHILE Partnumb # "  "
   @ 5,0 CLEAR
   Partnumb = SPACE(5)
```

```
@ 10,2 SAY "Enter data for goods received"
@ 12,4 SAY "Partnumber" GET Partnumb
READ
* - If a part number was entered , find it in
* - Master file
IF Partnumb # " "
    SEEK Partnumb
    DO CASE

        * - - - If part not found, warn user and
        * - - - - - try again
        CASE .NOT. FOUND()
            @ 12,23 SAY "No such part !!!"
            ? CHR(7)
            * - If found Append a new record to
            * - NewStock .dbf, and get
            * - rest of data.
        CASE FOUND()
            @ 12,25 SAY P_name
            SELECT B
            APPEND BLANK
            REPLACE Part_no WITH Partnumb
            REPLACE Date WITH T_Date
            REPLACE Vendor WITH A -> Vendor
            @ 14,2 SAY "Quantity" GET Qty
            @ 14,22 SAY "Price" GET COST PICT "99999.99"
            @ 16,2  SAY "Date" GET Date PICT "99/99/99"
            @ 16,22 SAY "Vendor" GET Vendor
            READ
            SELECT A
    ENDCASE
  ENDIF(Partnumb# "  ")
ENDDO(While user does not want to quit)
```

```
*----- Close files and return to New Stock
*----- menu.
CLOSE DATABASES
RETURN
```

والجزء الأول من البرنامج يقوم بفتح الملف الرئيسى (Master.dbf) لاختبار رقم الجزء الذى يتم إدخاله. كما يتم فتح ملف الإضافة (NewStock.dbf) لتخزين بيانات الأصناف الجديدة. ويتم ذلك من خلال السطور التالية :

```
SELECT A
USE Master INDEX Master
SELECT B
USE NewStock INDEX NewStock
SELECT A
```

والجزء الثانى يقوم بتكوين حلقة تكرارية لإدخال رقم الجزء. وذلك من خلال السطور التالية :

```
Partnumb = "x"
DO WHILE Partnumb # "  "
    @ 5,0 CLEAR
    Partnumb = SPACE(5)
    @ 10,2 SAY "Enter data for goods received"
    @ 12,4 SAY "Partnumber" GET Partnumb
    READ
```

والجزء الثالث يتم من خلاله البحث عن الرقم الذى قام المستخدم بإدخاله وذلك من خلال السطور التالية :

```
IF Partnumb # " "
   SEEK Partnumb
```

والجزء الرابع يتم من خلاله تحذير المستخدم فى حالة إدخال رقم جزء غير موجود. وذلك من خلال السطور التالية :

```
DO CASE
    CASE .NOT. FOUND()
        @ 12,23 SAY "No such part !!!"
        ? CHR(7)
```

والجزء الخامس يتم من خلاله عرض إسم الصنف الذى يتم إدخال رقمه على الشاشة وذلك فى حالة إدخال المستخدم لرقم موجود. كما يتم إضافة سجل خال فى نهاية ملف الإضافة (NewStock.dbf) حتى يتم إدخال البيانات الجديدة فيه. كما يتم ملء بيانات رقم الصنف والتاريخ وإسم البائع آليا وسؤال المستخدم عن الكمية والسعر وإسم البائع فى حالة تغييره ثم يتم التجهيز لإدخال صنف جديد. ويتم ذلك من خلال السطور التالية :

```
CASE FOUND()
        @ 12,25 SAY P_name
        SELECT B
        APPEND BLANK
        REPLACE Part_no WITH Partnumb
        REPLACE Date WITH T_Date
        REPLACE Vendor WITH A -> Vendor
        @ 14,2 SAY "Quantity" GET Qty
        @ 14,22 SAY "Price" GET COST PICT "99999.99"
        @ 16,2  SAY "Date" GET Date PICT "99/99/99"
        @ 16,22 SAY "Vendor" GET Vendor
        READ
        SELECT A
```

والجزء السادس يتم من خلاله إغلاق الملفات بعد انتهاء الحلقة التكرارية ثم العودة إلى قائمة الإضافة. وذلك من خلال السطور التالية :

```
            ENDCASE
        ENDIF(Partnumb# "  ")
    ENDDO(While user does not ask to quit)
    CLOSE DATABASES
    RETURN
```

١٦ - ٦ برنــامج تقاريــر الإضافـة (NewReps.prg)

يسمح هذا البرنامج للمستخدم بمراجعة بيانات الأصناف الجديدة فى صورة تقارير بناء على رقم الجزء (Part number) أو خلال فترة زمنية محددة. وتفيد هذه التقارير فى حل أى خلافات قد تنشأ مع البائعين (Vendors) عن طريق إمدادهم بالبيانات الدقيقة عن الأصناف وأسعارها.

ويتم تشغيل هذا البرنامج عندما يختار المستخدم الرقم (2) من قائمة الإضافة. انظر الشكل (١٦ - ٤).

```
New Stock System Menu          02/20/90    02:20:45

        1. Record new items
        2. Print new stock data
        3. Edit new stock data
        4. Return to main menu

  Enter choice (1-4) 0
```

شكل (١٦ - ٤)

فى هذه الحالة تظهر القائمة التالية على الشاشة.

1. By part number
2. By dates
3. Return to New Stock menu

وعند اختيار المستخدم للرقم (1) يظهر سؤال آخر عما إذا كان المطلوب طباعة التقرير أم عرضه فقط على الشاشة. ثم يظهر سؤال آخر عن رقم الجزء المطلوب عرض أو طباعة التقرير له. وعند كتابة المستخدم لهذا الرقم يظهر تقرير يوضح حركة الإضافة لهذا الصنف.

وعند اختيار المستخدم للرقم (2) من القائمة فإن هذا يعنى أنه يريد عرض حركة الأصناف من تاريخ معين إلى تاريخ آخر. وفى هذه الحالة يظهر على الشاشة سؤال عن تاريخ البداية وتاريخ النهاية كالآتى :

Enter start date : / /
Enter end date : / /

فيقوم المستخدم بكتابة المطلوب. وفى هذه الحالة تظهر كل بيانات حركة الإضافة التى تمت بين هذين التاريخين.

أنظر الشكل (١٦ - ٥) و الشكل (١٦ - ٦)

Page No 1 — **Inventory Items Received** — 02/20/90

Part Name	Part	Qty	Purchase Price	Date	Vendor Name
A-122	Shirt	20	500	02/20/90	Hasan
A-122	Shirt	35	750	02/01/90	Salem

الشكل(١٦ - ٥)

Page No 1 — **Inventory Items Received** — 02/20/90

Part Name	Part	Qty	Purchase Price	Date	Vendor Name
A-122	Shirt	30	750	02/01/90	Hasan
A-122	Shirt	20	500	02/20/90	Salem
A-121	Shoes	20	500	02/25/90	Hytham

شكل (١٦ - ٦)

ولكتابة البرنامج يتم أولا إنشاء صورة التقرير المطلوب مع ملاحظة أنه يمكن إنشاء صورة واحدة للتقرير واستخدامها فى حالة طلب التقرير بناء على رقم الجزء أو بناء على فترة زمنية محددة. ويتم ذلك باستخدام الأمر (CREATE) أو الأمر (MODIFY). وحيث أن التقرير يجب أن يتضمن بيانات من ملف الإضافة (NewStock.dbf) وكذلك بيانات من الملف الرئيسى (Master.dbf) ، لذلك يلزم عند إنشاء التقرير فتح ملف الإضافة وكذلك الملف الرئيسى والربط بينهما. ويتم ذلك من خلال الأوامر التالية :

```
CLEAR ALL
SELECT 1
USE NewStock
SELECT 2
USE Master INDEX Master
SELECT 1
SET RELATION TO Part_no INTO Master
MODIFY REPORT NewStock
```

مع ملاحظة أن هذه الأوامر تكتب من مشيرة النقطة (Dot Prompt) قبل بداية كتابة البرنامج.

وعندما تظهر الشاشة الخاصة بإنشاء التقرير يتم إدخال محتويات أعمدة التقرير (Columns) كما يظهر من الشكل التالى :

Column	Contents	Heading	Width	Decimals	Total
1	Part_no	Part	6		
2	B-->P_name	Part Name	15		
3	Qty	Qty	4	2	N
4	Cost	Purchase price	12	2	N
5	Date	Date	8		
6	Vendor	Vendor	25		

شكل (١٦٠ - ٧)

ويلاحظ هنا كتابة (B -> P_name) للحصول على إسم الجزء من الملف الرئيسى (Master.dbf).

١٦ - ٧ كتابة البرنامج

نظرا لأن هذا البرنامج شبيه ببرامج التقارير التى سبق كتابتها مع الملف الرئيسى (Master.dbf) وملف المبيعات (Sales.dbf) فسوف يكتفى هنا بكتابة البرنامج فقط ويمكن الرجوع إلى برامج التقارير السابقة لمتابعة شرحها. والبرنامج يتكون من السطور التالية :

```
* * * * * * * * * * * * * NewReps.prg
*   Print reports from the NewStock file
*   Called from NewStock menu, NMenu.prg .

* - - - - -  Open NewStock and Master databases.
SELECT 1
USE NewStock
SELECT 2
USE Master INDEX Master
* - - - - -  Set up relationship.
SELECT 1
SET RELATION TO Part_no INTO Master

* - - - - -  Start loop for menu
Repchoice = 0
DO WHILE Repchoice # 3
        CLEAR
        @ 2,1  SAY "New Stock Report options"
        @ 2,60 SAY DTOC(T_Date) + "  " + TIME()
        @ 3,0  SAY Uline
        ?
        ?
```

```
TEXT

        1. By part number
        2. By dates
        3. Return to New Stock menu

ENDTXT
@ 24,1 SAY "Enter choice(1-3)" ;
GET Repchoice PICT "9"
READ

* - - - - - If not exiting , ask about printer.
@ 3,0  CLEAR
STORE "  "  TO YN, Printer
IF Repchoice # 3
        @ 15,5 SAY "Send to printer?" GET YN PICT "!"
        READ
        * - - - - - Set up printer macro.
        IF YN = "Y"
                Printer = "TO PRINT"
        ENDIF
ENDIF

* - - - - - If not going to pritner, pause the
* - - - - - screen.
IF YN # "Y" .AND. Repchoice # 3
        ?
        ?
        WAIT "Press any key to return to the Repchoice menu"
ENDIF
ENDDO (while user does not request to exit)
* - - - - - when done , return to New Stock menu
```

```
SET RELATION TO
CLOSE DATABASES
RETURN
```

الفصل السابع عشر

برنامج تحديث البيانات

فى الفصول السابقة تم إنشاء ثلاثة أجزاء رئيسية من نظام المخازن للتحكم فى الملف الرئيسى (Master.dbf) وملف المبيعات (Sales.dbf) وملف الإضافة (NewStock.dbf). وفى هذا الفصل يتم إنشاء البرنامج المكمل للنظام والذى يمثل أهم جزء فيه. وهذا البرنامج يقوم بتحديث (Updating) لبيانات الملف الرئيسى (Master.dbf) بناء على البيانات الموجودة فى ملف المبيعات (Sales.dbf) وملف الإضافة (NewStock.dbf). وفى هذا الفصل أيضا يتم إنشاء برنامج تعديل ملف المبيعات وبرنامج تعديل ملف الإضافة.

١٧ - ١ برنامج تحديث الملف الرئيسى (Master.dbf)

يقوم هذا البرنامج بخصم كميات الأصناف الموجودة فى ملف المبيعات من الكميات الموجودة فى الملف الرئيسى لكل صنف تم البيع منه. كما يقوم بإضافة كميات الأصناف الموجودة فى ملف الإضافة إلى الكميات الموجودة فى الملف الرئيسى.

ويقوم هذا البرنامج أيضا بخصم الكميات الموجودة فى ملف الإضافة من الكميات الموجودة فى حقل تحت الطلب (On_order) فى الملف الرئيسى حيث أن هذه الأصناف تم إضافتها بالفعل. كما يقوم بتعديل سعر الشراء لهذه الأصناف (Purchase Price) بالسعر الموجود فى ملف الإضافة حيث أن هذا السعر يعتبر أحدث سعر للصنف. وأخيرا يقوم البرنامج بتعديل تاريخ آخر تحديث للبيانات بالتاريخ الموجود فى ملف الحركة المستخدم.

ويتم تشغيل هذا البرنامج عندما يختار المستخدم الرقم (4) من القائمة الرئيسية لبرنامج تشغيل الملف الرئيسى (Master.dbf). انظر الشكل (١٧ - ١)

```
Manage Master Inventroy        02/20/90    08:30:45

        1. Add new part numbers
        2. Print reports
        3. Make changes
        4. Update from sales and Newstock
        5. Return to main menu
 Enter choice (1-5)
```

شكل (١٧ - ١)

ولكتابة هذا البرنامج يتم أولا كتابة الخطوات الأولية (PSEUDOCODE).

١٧ - ١ - ١ كتابة الخطوات الأولية (PSEUDOCODE)

١ - يتم مسح الشاشة.
٢ - يتم عرض رسالة توضح للمستخدم أن الملف الرئيسى جارى تحديثه من ملف المبيعات.
٣ - يتم فتح ملف المبيعات والفهرس الخاص به.
٤ - يتم نسخ جميع سجلات ملف المبيعات التى لم يتم تحديثها (لم يتم ترحيل بياناتها) إلى ملف مؤقت (Temporary file).
٥ - يتم فتح الملف المؤقت.
٦ - يتم التأكد من وجود سجلات فى الملف المؤقت (التأكد أنه ليس فارغا).
٧ - فى حالة وجود سجلات فى الملف المؤقت يتم فتح الملف الرئيسى وملف الفهرس الخاص به.
٨ - يتم تحديث الملف الرئيسى من الملف المؤقت وذلك بخصم كميات الأصناف المباعة وتعديل حقل تاريخ آخر تحديث.
٩ - يتم إغلاق جميع الملفات.
١٠- يتم فتح ملف المبيعات (Sales.dbf).
١١- يتم إدخال القيمة (.T.) أى صحيح فى حقل الترحيل (Posted) لجميع سجلات ملف المبيعات لأن هذا يوضح أن جميع السجلات قد تم ترحيلها (Posted).
١٢- يتم إغلاق جميع الملفات.
١٣- يتم عرض رسالة للمستخدم توضح أن الملف الرئيسى جارى تحديثه من ملف الإضافة (NewStock.dbf).
١٤- يتم فتح ملف الإضافة وملف الفهرس الخاص به.
١٥- يتم نسخ جميع سجلات ملف الإضافة التى لم يتم تحديثها (لم يتم ترحيل بياناتها) إلى ملف مؤقت (Temporary file).
١٦- يتم فتح الملف المؤقت.
١٧- يتم التأكد من وجود سجلات فى الملف المؤقت.
١٨- فى حالة وجود سجلات فى الملف المؤقت يتم فتح الملف الرئيسى وملف الفهرس الخاص به.

١٩- يتم تحديث الملف الرئيسى من الملف المؤقت وذلك بإضافة كميات الأصناف المضافة وتعديل سعر الصنف وتاريخ آخر تحديث له وكذلك طرح كمية الصنف المضافة على المخزن من الكمية الموجودة فى حقل تحت الطلب (On_order).

٢٠- يتم إغلاق جميع الملفات.

٢١- يتم فتح ملف الإضافة (NewStock.dbf).

٢٢- يتم إدخال القيمة (.T.) أى صحيح فى حقل الترحيل (Posted) لجميع سجلات ملف الإضافة. حيث أن هذا يوضح أن جميع السجلات تم ترحيلها وذلك حتى لايعاد ترحيلها مرة ثانية.

٢٣- يتم إغلاق جميع الملفات.

٢٤- يتم الرجوع إلى قائمة برنامج تشغيل الملف الرئيسى (MMenu.prg).

١٧ - ١ - ٢ كتابة البرنامج

هذا البرنامج يتم تسميته (Updater.prg) ويتكون من السطور التالية :

```
* * * * * * * * * * * * * * * * * Updater.prg
*   Update the Master File from Sales and NewStock.
*   Called from Master menu , MMenu.prg.

* - - - - -  Ask user if sure abour updating.
YesNo = "Y"
@ 5,0  CLEAR
@ 15,4 SAY "Update Master file from Sales and" + ;
  "NewStock?(Y/N)" GET YesNo PICT "!"
READ
IF YesNo = "N"
   RETURN
ENDIF

* - - - - -  Display a message that Master is being
* - - - - -  updated from Sales file.
```

```
@ 5,0  CLEAR
@ 15,5 SAY "Updating from the Sales file ..."
USE Sales INDEX Sales

* - - - - -  Copy all unupdated records to Temp file.
COPY STRUCTURE TO Temp
COPY TO Temp FOR .NOT. Posted

* - - - - Make sure there are records in Temp
SELECT 2
USE Temp
IF RECCOUNT() > 0
   * Use Master file and index for updating
   SELECT 1
   USE Master INDEX Master

   * - - - - -  Update from the temporary sales file.
   UPDATE ON Part_no FROM Temp REPLACE Qty WITH ;
     Qty - Temp -> Qty , Date WITH Temp -> Date

   * - - - - -  USE original Sales file , make all
   * - - - - -  posted fields "True".
   CLOSE DATABASES
   USE Sales
   REPLACE ALL Posted WITH .T.
ENDIF(record count > 0)
CLOSE DATABASES

* - - - - -  Display a message that Master is bieng
* - - - - -  updated from the NewStock file.
@ 15,5 SAY "Updating from the NewStock file"
USE NewStock INDEX NewStock
```

```
* - - - - Copy all unupdated records to Temp file.
COPY STRUCTURE TO Temp
COPY TO Temp FOR .NOT. Posted

* - - - - -  Make sure there are records in temp.
SELECT 2
USE Temp
IF RECCOUNT() > 0
   * - - - USE Master file and index for updating
   SELECT 1
   USE Master INDEX Master
   * - - - Update from the temporary NewStock file.
   UPDATE ON Part_no FROM Temp ;
   REPLACE Qty WITH Qty + Temp -> Qty , Date WITH ;
   Temp -> Date, Cost WITH Temp -> Cost, ;
   On_order WITH On_order - Temp -> Qty

   * - - USE original NewStock file, make all posted
   * - - - - -  fields "True"
   CLOSE DATABASES
   USE NewStock
   REPLACE ALL Posted WITH .T.
ENDIF(record count > 0)

* - - - free up all work areas , and return to Master
* - - - menu.
CLOSE DATABASES
RETURN
```

والجزء الأول من البرنامج يبدأ بالتعريف باسم البرنامج (Updater.prg) ووظيفته ثم إسم البرنامج الذى قام باستدعائه (MMenu.prg) ثم يقوم بمسح الشاشة وعرض رسالة للمستخدم للتأكد من رغبته فى التحديث (Updating).

ثم يتم عرض رسالة توضح للمستخدم أن الملف الرئيسى (Master.dbf) جارى تحديثه من ملف المبيعات. ثم يقوم البرنامج بفتح ملف المبيعات (Sales.dbf) وملف الفهرس الخاص به. ثم يقوم بنسخ السجلات التى لم يتم ترحيلها إلى ملف مؤقت (Temp.dbf)...ويتم تنفيذ هذه الخطوات من خلال السطور التالية :

```
YesNo = "Y"
@ 5,0  CLEAR
@ 15,4 SAY "Update Master file from Sales and " + ;
"NewStock?(Y/N)" GET YesNo PICT "!"
READ
IF YesNo = "N"
  RETURN
ENDIF
* - - - - -  Display a message that Master is being
* - - - - -  updated from Sales file.
@ 5,0  CLEAR
@ 15,5 SAY "Updating from the Sales file ..."
USE Sales INDEX Sales
* - - - - -  Copy all unupdated records to Temp file.
COPY STRUCTURE TO Temp
COPY TO Temp FOR .NOT. Posted
```

والجزء الثانى من البرنامج يقوم بفتح الملف المؤقت للمبيعات(Temp.dbf) الذى يحتوى على كل السجلات المطلوب ترحيل بياناتها إلى الملف الرئيسى. ثم يقوم بالتأكد من وجود سجلات فى هذا الملف وذلك لتجنب التعامل مع ملف قاعدة بيانات فارغ لأن ذلك يمكن أن يؤدى إلى نتائج غير متوقعة. ثم يتم فتح الملف الرئيسى (Master.dbf) والفهرس الخاص به فى منطقة عمل أخرى. ويتم ذلك من خلال السطور التالية :

```
* Make sure there are records in Temp file.
SELECT 2
USE Temp
IF RECCOUNT()>0
```

```
* Use Master file and index for updating
SELECT 1
USE Master INDEX Master
```

والجزء الثالث من البرنامج يتم من خلاله إجراء عمليات الترحيل بخصم الكميات الموجودة فى الملف المؤقت (Temp -> Qty) من الكميات الموجودة فى الملف الرئيسى (Qty) ثم يتم تعديل تاريخ آخر تحديث للملف الرئيسى (Date) بالتاريخ الموجود فى ملف المبيعات. ويتم ذلك من خلال السطور التالية :

```
* - - - - - Update from the temporary sales file.
UPDATE ON Part_no FROM Temp REPLACE Qty WITH ;
Qty - Temp -> Qty , Date WITH Temp -> Date
```

والجزء الرابع يتم من خلاله تمييز السجلات التى تم ترحيلها حتى لايتم ترحيلها مرة ثانية. ويستخدم حقل الترحيل (Posted) فى هذه العملية حيث يتم إدخال القيمة (.T.) أى (True) صحيح فى هذا الحقل لجميع سجلات ملف المبيعات وهذا يعنى أن جميع البيانات قد تم ترحيلها إلى الملف الرئيسى. والسطور التالية توضح هذه العملية.

```
Close DATABASES
USE Sales
REPLACE ALL Posted WITH .T.
```

ثم يتم إغلاق جميع الملفات حتى يمكن البدء فى تحديث الملف الرئيسى من ملف الإضافة. ويتم ذلك من خلال السطور التالية :

```
ENDIF(record count > 0)
CLOSE DATABASES
```

والجزء الخامس يتم من خلاله تنفيذ عملية التحديث من خلال ملف الإضافة (NewStock.dbf) بنفس الطريقة السابقة حيث يتم أولا عرض رسالة على

المستخدم توضح له إجراء عملية التحديث ثم يتم فتح ملف الإضافة والفهرس الخاص به. ثم يتم نسخ جميع السجلات التى لم يتم تحديثها (ترحيلها من ملف الإضافة) إلى ملف مؤقت (Temp). وحيث أنه قد سبق كتابة الأمر (SET SAFETY OFF) فى البرنامج الرئيسى لبرنامج المخازن(IMenu.prg) فإن عملية نسخ السجلات تتم دون ظهور الرسالة التحذيرية المعتادة فى حالة النسخ فوق ملف سابق. والسطور التالية توضح تنفيذ هذه الخطوات.

```
@ 15,5 SAY "Updating from the NewStock file"
USE NewStock INDEX NewStock
* - - - - -  Copy unupdated records to Temp file.
COPY STRUCTURE TO Temp
COPY TO Temp FOR .NOT. Posted
```

ثم يتم التأكد من وجود سجلات فى الملف المؤقت (Temp) وذلك من خلال السطور التالية :

```
SELECT 2
USE Temp
IF RECCOUNT() > 0
```

ثم يتم فتح الملف الرئيسى (Master.dbf) وملف الفهرس الخاص به من خلال السطور التالية :

```
SELECT 1
USE Master INDEX Master
```

ثم يقوم البرنامج بعد ذلك بترحيل القيم الموجودة فى حقول الملف المؤقت (Temp) إلى الملف الرئيسى (Master). وذلك بإضافة الكمية الموجودة فى حقل الكمية (Temp -> Qty) على الكمية الموجودة فى الملف الرئيس(Qty). كما يتم استبدال السعر بالسعر الموجود فى الملف المؤقت (Temp->cost) وكذلك تاريخ آخر تحديث. كما يتم طرح الكمية (Temp->Qty) من الكمية الموجودة فى حقل تحت الطلب (On_order). والسطور التالية توضح تنفيذ هذه الخطوات.

```
* - - - - -  Update from the temporary NewStock file.
UPDATE ON Part_no FROM Temp ;
REPLACE Qty WITH Qty + Temp -> Qty , Date WITH ;
Temp -> Date, Cost WITH Temp -> Cost, ;
On_order WITH On_order - Temp -> Qty
```

وفى الجزء الأخير من البرنامج يتم إدخال القيمة (.T.) فى كل حقول الترحيل (Posted) حتى لايتم ترحيلها مرة ثانية ثم يتم إغلاق جميع الملفات والعودة إلى قائمة تشغيل الملف الرئيسى (MMenu.prg). والسطور التالية توضح هذه الخطوات :

```
   * - - - USE original NewStock file, make all
   * - - - posted fields "True"
   CLOSE DATABASES
   USE NewStock
   REPLACE ALL Posted WITH .T.
ENDIF(record count > 0)

* - - - - -  free up all work areas , and return to
* - - - - -  Master menu.
CLOSE DATABASES
RETURN
```

١٧ - ٢ برنامج تصحيح ملف المبيعات (SalEdit.prg)

كما يلاحظ من عرض البرامج الفرعية المختلفة المكونة لبرنامج المخازن فقد تم تأجيل شرح البرنامج الخاص بتصحيح ملف المبيعات (SalEdit.prg) إلى هذا الفصل. والسبب فى ذلك أن التصحيح بعد عملية الترحيل يؤدى إلى عدم مطابقة بيانات الملف الرئيسى (Master.dbf) للواقع. فمثلا لو فرضنا أن المستخدم باع عشرة طابعات ولكنه كتب فى حقل الكمية الخاص بهذا الصنف عشرين بدلا من عشرة. فإذا قام بتعديل هذا الصنف فى ملف المبيعات بعد ترحيله إلى الملف الرئيسى (Master) فإن هذا التعديل لايظهر فى الملف الرئيسى وبالتالى لايصبح الملف الرئيسى معبرا عن الكمية الصحيحة فى المخزن. ولذلك فإن

برنامج تصحيح ملف المبيعات يسمح للمستخدم باجراء أى تعديلات يريدها للملف المبيعات وفى نفس الوقت يقوم آليا بإدخال هذه التعديلات على الملف الرئيسى. والخطوات المنطقية لتنفيذ هذه العملية تتلخص فى الآتى :

١- السماح للمستخدم بتعديل أى بيانات سجل معين.
٢- إذا قام المستخدم بتعديل بيان رقم جزء (Part number) فى ملف المبيعات يقوم البرنامج بإضافة الكمية المباعة من هذا الصنف على رقم الجزء السابق وطرحها من رقم الجزء الجديد فى الملف الرئيسى.
٣- إذا قام المستخدم بتعديل الكمية المباعة فى ملف المبيعات يتم حساب الفرق بين الكمية التى سبق إدخالها والكمية الجديدة. ثم يتم طرح هذه الكمية من الكمية الموجودة فى الملف الرئيسى.

وليست هناك حاجة لتعديل تاريخ آخر تحديث إذا قام المستخدم بتعديله فى ملف المبيعات لأنه فى جميع الأحوال يمثل تاريخ التحديث (Updating) وليس التعديل (Editing).

وهذا البرنامج عند تشغيله يؤدى إلى عرض الرسالة التالية على الشاشة.

Enter Invoice number to edit (0 if none) :

وعندما يكتب المستخدم رقم الفاتورة المطلوب التعديل فيها فإن البرنامج يعرض كل الحركة الخاصة بهذه الفاتورة متضمنة بيانات الأصناف المختلفة. وفى هذه الحالة يقوم المستخدم باختيار أحد هذه الأصناف لتعديل بياناته فتظهر شاشة خاصة بالتعديل متضمنة بيانات هذا الصنف لتعديلها أو مسحها. انظر الشكل (١٧ - ٢).

ويستطيع المستخدم تحريك المؤشر إلى أى بيان خاص بهذا الصنف وتعديله. كما يمكنه كتابة (Y) أمام السؤال الذى يظهر أسفل الشاشة كالآتى :

Delete this record ? (Y/N)

وذلك عندما يريد إلغاء هذا الصنف.

ويقوم البرنامج بعد ذلك بإجراء التعديلات المطلوبة على الملف الرئيسى.

Edit Sales Transactions	02/20/90	10:50:30

Invoice Number : ▒▒▒▒ Date : ▒▒
Part Number : ▒▒▒
Clerk : ▒▒▒ Customer : ▒▒▒
Quantity : ▒▒▒ Selling Price: ▒▒▒

Delete this record (Y/N) ▒

شكل (١٧ - ٢)

١٧ - ٢ - ١ الخطوات الأولية (PSEUDOCODE)

يتم كتابة الخطوات الأولية للبرنامج كالآتى :

١ - يتم إنشاء متغيرات الذاكرة.
٢ - يتم تكوين حلقة تكرارية لإدخال رقم الفاتورة.
٣ - يتم فتح ملف المبيعات(Sales.dbf) وملف الفهرس الخاص به (Sales.ndx).
٤ - يتم حساب عدد السجلات التى تشترك فى رقم الفاتورة الذى يدخله المستخدم.
٥ - إذا لم يتم العثور على أى سجل بنفس رقم الفاتورة يتم تحذير المستخدم والسماح له بالمحاولة مرة ثانية مع رقم فاتورة آخر.
٦ - إذا كانت هناك عدة سجلات تشترك فى رقم الفاتورة. يتم عرض بياناتها للمستخدم وسؤاله عن رقم السجل المطلوب تعديل بياناته. وإذا كان هناك سجل واحد يتم الذهاب إليه.
٧ - يتم الذهاب إلى السجل المطلوب تعديل بياناته فى ملف المبيعات وتخزين القيم الموجودة فيه فى متغيرات ذاكرة.

٨ - يتم عرض بيانات السجل من خلال شاشة تصحيح والسماح للمستخدم بتصحيح أى بيانات فى السجل.

٩ - بعد انتهاء عملية التصحيح يتم ضبط بيانات الملف الرئيسى.

١٠- إذا كان السجل قد تم ترحيله (Posted) ثم قام المستخدم بمسحه يتم تجهيزه للمسح من الملف الرئيسى.

١١- إذا كان السجل قد تم ترحيله وقام المستخدم بتعديل رقم الجزء (Part number) يتم الذهاب إلى رقم الجزء القديم فى الملف الرئيسى (Master) وإضافة الكمية التى سبق إنقاصها منه. كما يتم الذهاب إلى رقم الجزء الجديد فى الملف الرئيسى وطرح هذه الكمية منه.

١٢- إذا كان السجل قد تم ترحيله (Posted) وقام المستخدم بتعديل الكمية يتم حساب الفرق بين الكمية السابقة والكمية الجديدة.

١٣- يتم الذهاب إلى رقم الجزء فى الملف الرئيسى وطرح الفرق من الكمية الموجودة (Quantity).

١٤- السماح للمستخدم بتكرار هذه العملية لأى جزء آخر.

١٥- إذا كانت هناك سجلات قد سبق تجهيزها للمسح يتم إنهاء عملية المسح باستخدام الأمر (PACK).

١٦- يتم الرجوع إلى برنامج تشغيل ملف المبيعات (SMenu.prg).

١٧ - ٢ - ٢ كتابة البرنامج

يتم كتابة سطور البرنامج كالآتى :

```
* * * * * * * * * * * * * * * * * * SalEdit.prg
*   Edit the Sales File and update Master.
*   Called from Sales Menu , SMenu.prg.

*   Set up memory variables.
No_Dels = 0
Search = 1

* - - - - - Set up loop for invoice numbers .
DO WHILE Search # 0
   CLEAR
```

```
@ 2,1  SAY "Edit Sales Transactions"
@ 2,60 SAY DTOC(T_Date) + "   " + TIME()
@ 3,0  SAY Uline
?
?
@ 15,5 SAY "Enter invoice number (0 if none) : ;
" GET Search PICT "9999"
READ

* - - - - If user does not request exit, continue
* - - - - - with edit.
IF Search > 0
USE Sales INDEX Sales
* - - - Count records with that invoice number.
COUNT FOR invoice_no = Search TO Howmany
DO CASE
   * - - - If invoice not found , warn user.
   CASE Howmany = 0
           @ 24,1 SAY "No such invoice number"
           ? CHR(7)
   * - - - - -If invoice number found , proceed.
   CASE Howmany > 0
           * - - If several records have this invoice
           * - - number , dispaly them and get the
           * - - required record number
           IF Howmany > 1
              @ 5,1  CLEAR
              ? "RecNo  Part #  Qty  Price Clerk"
              ?? " Customer   Date"
              ?
              LIST FOR Invoice_no = Search Part_no , ;
              Qty, Price , Clerk, Customer , Date
              ?
```

```
    INPUT "Edit which record:" TO RecNo
    GOTO RecNo
* - - - Otherwise, just go to the record.
ELSE
    LOCATE FOR Invoice_no = Search
ENDIF

* Store original field values to variables.
Old_Part = Part_no
Old_Qty  = Qty
* - - Display data on edit screen and allow
* - - edit.
@ 5,0 CLEAR
Deleted = " "
@ 7,1  SAY "Invoice Number" GET Invoice_no
@ 7,44 SAY "Date" GET Date PICT "99/99/99"
@ 10,1 SAY "Part Number" GET Part_no PICT ;
        "!!!!!"
@ 12,40 SAY "Customer" GET Customer
@ 15,1 SAY "Qauntity" GET Qty
@ 15,20 SAY "Selling price" GET Price
@ 24,1 SAY "Delete this record? (Y/N)" ;
GET Deleted PICT "!"
READ
* - - - After editing , adjust Master file.
DO CASE
        * - - - If transaction to be deleted ,
        * - - - delete it and add its quantity
        * - - - back to the Master file.
        CASE Posted .AND. Deleted = "Y"
                DELETE
                No_Dels = No_Dels + 1
```

```
                USE Master INDEX Master
                SEEK Old_Part
                IF FOUND()
                        REPLACE Qty WITH Qty + Old_Qty
                ENDIF
                * - If Part number changed , add
                * - quantity to old part number,
                * - and subtract it from new part number.
        CASE Posted .AND. Part_no # Old_Part
                New_Qty = Qty
                New_Part = Part_no
                USE Master INDEX Master
                SEEK Old_Part
                IF FOUND()
                        REPLACE Qty WITH Qty + Old_Qty
                ENDIF
        * - -If quantity changed
        * - -adjust the Master file quantity
        CASE Posted .AND. Qty # Old_Qty
                Diff = Qty - Old_Qty
                USE Master INDEX Master
                SEEK Old_Part
                IF FOUND()
                        REPLACE Qty WITH Qty - Diff
                ENDIF
                SEEK New_Part
                IF FOUND()
                        REPLACE Qty WITH Qty - New_Qty
                ENDIF
    ENDCASE(adjustments after editing)
ENDCASE(Howmany > 0)
ENDIF(Search > 0)
```

```
ENDDO(while still editing)

* - - - -If there are records to be deleted , Pack the
* - - - -sales database file.
IF No_Dels > 0
        @ 5,0 CLEAR
         ? "Deleting unwanted records from the sales file.."
        USE Sales INDEX Sales
        PACK
ENDIF

* - - - - -  Return to Sales menu.
CLOSE DATABASES
RETURN
```

والبرنامج يبدأ بإنشاء متغير الذاكرة (No_Dels) لتخزين عدد السجلات المطلوب مسحها وكذلك إنشاء متغير الذاكرة (Search) لتخزين رقم الفاتورة التى يريد المستخدم تعديلها. ويتم ذلك من خلال السطور التالية :

```
    No_Dels = 0
    Search = 1
```

ثم يتم تكوين حلقة تكرارية لسؤال المستخدم عن رقم الفاتورة المطلوب تعديلها. وذلك من خلال السطور التالية :

```
DO WHILE Search # 0
CLEAR
@ 2,1  SAY "Edit Sales Transactions"
@ 2,60 SAY DTOC(T_Date) + "   " + TIME()
@ 3,0  SAY Uline
?
?
```

```
@ 15,5 SAY "Enter invoice number  (0 if none) :";
GET Search PICT "9999"
READ
```

ثم يتم فتح ملف المبيعات (Sales.dbf) والفهرس الخاص به للبحث عن رقم الفاتورة الذى أدخله المستخدم (فى حالة إدخال هذا الرقم). ويتم حصر عدد السجلات التى تشترك فى هذا الرقم وتخزين هذا العدد فى المتغير (Howmany) ويتم ذلك من خلال السطور التالية :

```
IF Search > 0
USE Sales INDEX Sales
* - - - - - Count records with that invoice number.
COUNT FOR invoice_no = Search TO Howmany
```

إذا لم يكن هناك أى سجلات بنفس رقم الفاتورة يتم تحذير المستخدم وذلك من خلال السطور التالية :

```
DO CASE
   * - - - - - If invoice not found , warn user.
   CASE Howmany = 0

   @ 24,1 SAY "No such invoice number"
   ? CHR(7)
```

إذا كان هناك عدة سجلات لها نفس رقم الفاتورة يقوم البرنامج بعرض بيانات هذه السجلات حتى يقوم المستخدم بتحديد رقم السجل المطلوب تعديله ثم يتم الذهاب إلى هذا السجل لتعديله. ويتم ذلك من خلال السطور التالية :

```
CASE Homany > 0
   * - - - If several records have this invoice
   * - - - number , dispaly them and get the
   * - - - required record number
```

```
IF Howmany > 1
   @ 5,1  CLEAR
   ? "RecNo  Part #  Qty  Price  Clerk"
   ?? " Customer   Date"
   ?
   LIST FOR Invoice_no = Search Part_no , ;
   Qty, Price , Clerk, Customer , Date
   ?
   INPUT "Edit which record:" TO RecNo
   GOTO RecNo
```

إذا كان هناك سجل واحد برقم الفاتورة الذى تم إدخاله يتم الذهاب إلى هذا السجل وذلك كالآتى :

```
ELSE
LOCATE FOR Invoice_no = Search
ENDIF
```

وفى الجزء التالى يتم تخزين رقم الجزء الأصلى والكمية الأصلية الموجودة فى السجل الذى تم اختياره فى متغيرات ذاكرة قبل أن يقوم المستخدم بتعديلها وذلك حتى يمكن اختبار هذه الحقول بعد ذلك. ويتم ذلك من خلال السطور التالية :

```
Old_Part = Part_no
Old_Qty  = Qty
```

وفى الجزء التالى يتم إنشاء متغير الذاكرة (Deleted) لتخزين رغبة المستخدم فى مسح السجل أو عدم مسحه ثم يتم تصميم شاشة لتصحيح بيانات السجل. ويتم ذلك من خلال السطور التالية :

```
@ 5,0 CLEAR
Deleted = "  "
@ 7,1  SAY "Invoice Number" GET Invoice_no
```

```
@ 7,44 SAY "Date" GET Date PICT "99/99/99"
@ 10,1 SAY "Part Number" GET Part_no ;
 PICT "!!!!!"
@ 12,1 SAY "Clerk" GET Clerk
@ 12,40 SAY "Customer" GET Customer
@ 15,1 SAY "Qauntity" GET Qty
@ 15,20 SAY "Selling price" GET Price
@ 24,1 SAY "Delete this record? (Y/N)" GET ;
 Deleted PICT "!"
READ
```

وفى الجزء التالى من البرنامج يتم اختبار هذا السجل بعد تعديله بواسطة المستخدم. فإذا كان السجل قد تم ترحيله (Posted) وطلب المستخدم مسحه (Deleted = "Y") يتم تجهيزه للمسح بواسطة الأمر (DELETE) ثم يتم زيادة عدد السجلات المطلوب مسحها (No_Dels) بواحد.

ثم يقوم البرنامج بعد ذلك بفتح الملف الرئيسى (Master) وملف الفهرس الخاص به للبحث عن هذا السجل المطلوب مسحه وإضافة الكمية التى عدلها المستخدم مرة ثانية. ويتم ذلك من خلال السطور التالية :

```
DO CASE
   * - - - - -  If transaction to be deleted ,
   * - - - - -  delete it and add its quantity
   * - - - - -  back to the Master file.
   CASE Posted .AND. Deleted = "Y"
        DELETE
        No_Dels = No_Dels + 1
        USE Master INDEX Master
        SEEK Old_Part
        IF FOUND()
              REPLACE Qty WITH Qty + Old_Qty
        ENDIF
```

وفى الجزء التالى يتم اختبار حالة ثانية فى السجل وهى عندما يقوم المستخدم بتعديل رقم الجزء. وذلك عندما يكتشف مثلا أنه أدخل رقم أحد الأجزاء خطأ عند كتابة فاتورة البيع. وفى هذه الحالة تكون كمية الصنف (Quantity) الخاصة بهذا الجزء قد نقصت فى الملف الرئيسى (Master) فى حين تظل كمية الصنف الذى كان يجب إدخال رقمه كما هى. ولذلك يقوم البرنامج بإضافة الكمية التى سبق خصمها من الصنف فى الملف الرئيسى إلى الجزء القديم (Old_Part) وخصمها من كمية الصنف فى الجزء الجديد. ويتم ذلك من خلال السطور التالية :

```
CASE Posted .AND. Part_no # Old_Part
        New_Qty = Qty
        New_Part = Part_no
        USE Master INDEX Master
        SEEK Old_Part
        IF FOUND()
                REPLACE Qty WITH Qty + Old_Qty
        ENDIF
```

والجزء التالى يعالج الحالة الثالثة وهى عندما يعدل المستخدم الكمية (Quantity) فقط دون تعديل رقم الجزء. فى هذه الحالة يقوم البرنامج بتعديل كمية هذا الجزء فى الملف الرئيسى (Master) وذلك بخصم الكمية القديمة من الكمية الجديدة التى عدلها المستخدم وتخزين هذا الفرق فى المتغير (DIFF). ثم يتم طرح هذا الفرق (DIFF) من الكمية الحالية الموجودة فى الملف الرئيسى ويتم ذلك من خلال السطور التالية :

```
CASE Posted .AND. Qty # Old_Qty
        Diff = Qty - Old_Qty
        USE Master INDEX Master
        SEEK Old_Part
        IF FOUND()
                REPLACE Qty WITH Qty - Diff
        ENDIF
ENDCASE(adjustments after editing)
```

```
ENDCASE(Howmany > 0)
ENDIF(Search > 0)
ENDDO(while still editing)
```

وفى الجزء التالى يتم اختبار المتغير (No_Dels) لمعرفة ما إذا كانت هناك سجلات مطلوب مسحها أم لا. فإذا كانت هناك سجلات مطلوب مسحها يتم استخدام الأمر (PACK) فى مسحها نهائيا من ملف المبيعات (Sales.dbf). ويتم ذلك من خلال السطور التالية :

```
IF NO_Dels > 0
        @ 5,0 CLEAR
        ? "Deleting unwanted records from the sales file.."
        USE Sales INDEX Sales
         PACK
ENDIF
```

وفى نهاية البرنامج يتم إغلاق الملفات والعودة إلى برنامج المبيعات (SMenu.prg).

١٧ - ٣ برنامج تصحيح ملف الإضافة (NewEd.prg)

يتم تشغيل هذا البرنامج عندما يختار المستخدم الاختيار رقم (3) فى قائمة برنامج الإضافة. انظر الشكل (١٧ - ٣).

```
New Stock System Menu          02/20/90    08:30:45

        1. Record new items
        2. Print new stock data
        3. Edit new stock data
        4. Return to main menu

  Enter choice (1 - 4)
```

شكل (١٧ - ٣)

وهذا البرنامج مثل البرنامج السابق يسمح للمستخدم بتعديل سجلات إضافة الأصناف التى سبق إدخالها بحيث يتم تعديل الملف الرئيسى (Master) بناء على هذه التعديلات إذا كانت السجلات قد سبق ترحيلها (Posted). وهو لايختلف فى تركيبه عن البرنامج السابق ولكنه يختلف عنه فى الحاجة إلى تعديل حقل سعر الصنف (Cost) وحقل (On_Order) بالإضافة إلى حقل الكمية (Qty). كما أن البرنامج يستخدم رقم الجزء فى البحث عن الصنف المطلوب فى ملف الإضافة (NewStock.dbf) وليس رقم الفاتورة مثل ملف المبيعات. ويتكون البرنامج من السطور التالية :

```
****************** NewEdit.prg
*   Edit the NewStock File and update Master.
*   Called from NMenu.prg.
*   Set up memory variables.
No_Dels = 0
Search = "1"

* - - - - - Get part number for data to edit.
DO WHILE Search # "0"
        Use NewStock INDEX NewStock
        CLEAR
        @ 2,1  SAY "Edit NewStock Transactions"
        @ 2,60 SAY DTOC(T_Date) + "   " + TIME()
         @ 3,0  SAY Uline
         ?
        ACCEPT "Enter part number to edit (0 if none)" TO Search
        Search = UPPER(Search)
        * - - - If user did not request exit, continue.
         IF Search # "0"
                * - - Count the records with that part number
                SEEK Search
                COUNT WHILE Part_no = Search TO Howmany
                DO CASE
                        * - - - If part not found , warn user.
```

```
CASE Howmany = 0
        ? "  No such part number"
        ? CHR(7)

* - - - If part number found , proceed.
CASE Howmany > 0
        * - -  If several records have that part
        * - -  number , dispaly them and get the
        * - -  required record
        IF Howmany > 1
        CLEAR
            SEEK Search
            LIST WHILE Part_no = Search Part_no , ;
              Qty, Cost, Date, Vendor
           ?
           INPUT "Edit which record(enter ;
           record number)?" TO RecNo
           GOTO RecNo
        ELSE
          SEEK Search
          RecNo = RECNO()
        ENDIF(Howmany > 1)

        * Store original field values to variables
        Old_Part = Part_no
        Old_Qty = Qty
        Old_Cost = Cost
        *- - Display edit screen and allow edit.
        CLEAR
        Deleted = "  "
        @ 1,1  SAY "Edit NewStock Transaction"
        @ 3,1  SAY "Part Number" ;
```

```
   GET Part_no PICT "!!!!!"
@ 5,1  SAY "Quantity" GET Qty
@ 5,20 SAY "Purchase Price" GET Cost
@ 7,1  SAY "Date" GET Date PICT "99/99/99"
@ 7,15 SAY "Vendor" GET Vendor
@ 9,2  SAY "Delete this record? (Y/N)" ;
       GET Deleted PICT "!"
READ
* - After editing , adjust Master file.
* - -  First handle change in cost.
IF Cost # Old_Cost
       New_Part = Part_no
       New_Cost = Cost
       USE Mater INDEX Master
       SEEK New_Part
       IF FOUND()
              REPLACE Cost WITH New_Cost
              USE NewStock INDEX NewStock
              GOTO RecNo
       ENDIF
ENDIF (Cost # Old_Cost)
DO CASE
    * - - -if new stock transaction to
    * - - -be deleted, subtract its
    * - - -quantity from the Master
    * - - -file and add it to the
    * - - -On_order field.
    CASE Posted .AND. Deleted = "Y"
       DELETE
       No_Dels = No_Dels + 1
       USE Master INDEX Master
       SEEK Old_Part
```

```
    IF FOUND()
         REPLACE Qty WITH Qty +  Old_Qty
         REPLACE On_order WITH  On_order ;
         + Old_Qty
    ENDIF
* - - If Part number changed,
* - - subtract quantity from the old
* - - part number, and add to the
* - - new . Do the opposit for the
* - - On_order field.
CASE Posted .AND. Part_no # Old_Part
    New_Qty = Qty
    New_Part = Part_no
    USE Master INDEX Master
     SEEK Old_Part
    IF FOUND()
         REPLACE Qty WITH Qty - Old_Qty
         REPLACE On_order WITH On_order ;
         + Old_Qty
    ENDIF
     SEEK New_Part
    IF FOUND()
         REPLACE Qty WITH Qty + New_Qty
         REPLACE On_order WITH On_order;
         - New_Qty
    ENDIF
* - - If user just changed the
* - - quantity , adjust the
* - - Master file quantity.
CASE Posted .AND. Qty # Old_Qty
    Diff = Qty - Old_Qty
    USE Master INDEX Master
```

```
                    SEEK Old_Part
                    IF FOUND()
                          REPLACE Qty WITH Qty + Diff
                          REPLACE On_order WITH ;
                          On_order - Diff
                    ENDIF
                ENDCASE
   ENDCASE
  ENDIF(Search # 0).
ENDDO(while Search # 0)

* - - -  If records have been deleted , Pack the New
* - - - - Stock data file.
IF NO_Dels > 0
      CLEAR
      ? "Deleting unwanted records from NewStock file"
      USE NewStock INDEX NewStock
      PACK
ENDIF

* - - - - -  Done , Return to NewStock Menu.
CLOSE DATABASES
RETURN
```

والبرنامج يبدأ بإنشاء متغيرات الذاكرة مثل البرنامج السابق تماما ولكن يلاحظ أن المتغير (Search) يتم إنشاؤه كمتغير حرفي. وذلك لأن البحث سوف يتم عن رقم الجزء وليس رقم الفاتورة مثل البرنامج السابق ورقم الجزء يمكن أن يحتوى على أرقام وحروف. ولذلك تستخدم الدالة (UPPER) فى تحويل رقم الجزء الذى يدخله المستخدم إلى حروف كبيرة حتى يماثل الفهرس. والسطور التالية توضح هذا الجزء.

```
No_Dels = 0
Search = "1"
```

```
* - - - - - Get part number for data to edit.
DO WHILE Search # "0"
   Use NewStock INDEX NewStock
   CLEAR
   @ 2,1  SAY "Edit NewStock Transactions"
   @ 2,60 SAY DTOC(T_Date) + "    " + TIME()
   @ 3,0  SAY Uline
   ?
   ACCEPT "Enter part number to edit(0 if none)" TO ;
   Search
   Search = UPPER(Search)
```

ثم يتم البحث عن رقم الجزء الذى أدخله المستخدم وحصر عدد السجلات التى لها هذا الرقم وتخزين هذا العدد فى المتغير (Howmany). ويتم ذلك من خلال السطور التالية :

```
IF Search # "0"
* - - - - - Count the records with that part number
SEEK Search
COUNT WHILE Part_no = Search TO Howmany
```

وإذا لم يتم العثور على هذا الرقم يتم تحذير المستخدم وذلك كالآتى :

```
DO CASE
   * - - - - - If part not found , warn user.
   CASE Howmany = 0
   ? "  No such part number"
   ? CHR(7)
```

وإذا كان هناك عدة سجلات لها نفس رقم الجزء المطلوب يتم عرض بيانات هذه السجلات وسؤال المستخدم عن رقم السجل المطلوب. ويتم ذلك من خلال السطور التالية :

```
IF Howmany > 1
```

```
CLEAR
SEEK Search
LIST WHILE Part_no = Search Part_no , ;
Qty, Cost, Date, Vendor
?
INPUT "Edit which record(enter record number) ?" TO RecNo
GOTO RecNo
```

وإذا كان هناك سجل واحد يحتوى على رقم الجزء المطلوب يتم الذهاب إلى هذا السجل وتخزين رقم السجل فى متغير الذاكرة (RecNo) ويتم ذلك من خلال السطور التالية :

```
ELSE
  SEEK Search
  RecNo = RECNO()
ENDIF(Howmany > 1)
```

وفى الجزء التالى يتم تخزين محتويات بعض حقول السجل المطلوب تعديله فى متغيرات ذاكرة حتى يمكن استخدامها بعد ذلك فى تعديل الملف الرئيسى. ويتم ذلك من خلال السطور التالية :

```
old_part = part_no
old_Qty  = Qty
old_Cost = Cost
```

وفى الجزء التالى يقوم البرنامج بعرض بيانات سجل الإضافة للمستخدم حتى يمكنه تعديل الحقول المطلوبة أو مسح السجل من ملف الإضافة (NewStock.dbf) حسب الحاجة. ويتم ذلك من خلال السطور التالية :

```
CLEAR
Deleted = " "
@ 1,1  SAY "Edit NewStock Transaction"
@ 3,1  SAY "Part Number" GET Part_no  PICT "!!!!!"
@ 5,1  SAY "Quantity" GET Qty
```

```
@ 5,20 SAY "Purchase Price" GET Cost
@ 7,1  SAY "Date" GET Date PICT "99/99/99"
@ 7,15 SAY "Vendor" GET Vendor
@ 9,2  SAY "Delete this record? (Y/N)" GET  Deleted PICT "!"
READ
```

وفى الجزء التالى يقوم البرنامج بضبط البيانات فى الملف الرئيسى (Master.dbf) بناء على البيانات التى تم تعديلها فى ملف الإضافة (NewStock.dbf). فإذا كان المستخدم عدل سعر الجزء (Cost) يتم تعديله فى الملف الرئيسى. وذلك من خلال السطور التالية :

```
IF Cost # Old_Cost
  New_Part = Part_no
  New_Cost = Cost
  USE Mater INDEX Master
  SEEK New_Part
  IF FOUND()
    REPLACE Cost WITH New_Cost
    USE NewStock INDEX NewStock
    GOTO RecNo
  ENDIF
ENDIF(Cost # Old_Cost)
```

وفى الجزء التالى يتم اختبار حالة أخرى عندما يطلب المستخدم مسح هذا السجل (Deleted = "Y") فإن البرنامج يقوم بتجهيزه للمسح باستخدام الأمر (DELETE) ثم يقوم بفتح الملف الرئيسى والفهرس الخاص به. ويتم البحث عن هذا السجل فإذا كان موجودا فى الملف الرئيسى يتم خصم الكمية التى سبق إضافتها من الملف الرئيسى كما يتم إضافة هذه الكمية إلى الكمية تحت الطلب (On_order). وذلك من خلال السطور التالية :

```
DO CASE
CASE Posted .AND. Deleted = "Y"
```

```
DELETE
No_Dels = No_Dels + 1
USE Master INDEX Master
SEEK Old_Part
IF FOUND()
   REPLACE Qty WITH Qty - Old_Qty
   REPLACE On_order WITH ;
   On_order + Old_Qty
ENDIF
```

والجزء التالى يعالج حالة أخرى عندما يقوم المستخدم بتعديل رقم الجزء الذى سبق الإضافة عليه. وفى هذه الحالة يقوم البرنامج بخصم الكمية المضافة من الجزء السابق وإضافتها إلى الجزء الجديد. كما يتم إضافة هذه الكمية إلى الكمية تحت الطلب ويتم ذلك من خلال السطور التالية :

```
CASE Posted .AND. Part_no # Old_Part
   New_Qty = Qty
   New_Part = Part_no
   USE Master INDEX Master
   SEEK Old_Part
   IF FOUND()
      REPLACE Qty WITH Qty - Old_Qty
      REPLACE On_order WITH On_order + Old_Qty
   ENDIF
   SEEK New_Part
   IF FOUND()
      REPLACE Qty WITH Qty - New_Qty
      REPLACE On_order WITH On_order  - New_Qty
   ENDIF
```

والجزء التالى يعالج الحالة الأخيرة عندما يقوم المستخدم بتعديل الكمية التى سبق إضافتها. فى هذه الحالة يتم ضبط الكمية فى الملف الرئيسى بناء على الفرق بين الكمية

الأولى والكمية الجديدة بعد التعديل. والسطور التالية توضح هذه العملية :

```
CASE Posted .AND. Qty # Old_Qty
   Diff = Qty - Old_Qty
   USE Master INDEX Master
   SEEK Old_Part
   IF FOUND()
      REPLACE Qty WITH Qty + Diff
      REPLACE On_order WITH  On_order - Diff
   ENDIF
ENDCASE
ENDCASE
ENDIF(Search # 0)
ENDDO(while Search # 0)
```

وفى الجزء التالى يتم مسح السجلات التى سبق تجهيزها للمسح وذلك باستخدام الأمر (PACK). ثم يتم إغلاق الملفات والعودة إلى برنامج الإضافة (NMenu.prg). والسطور التالية توضح هذه العملية :

```
IF NO_Dels > 0
  CLEAR
  ? "Deleting unwanted records from the NewStock file"
  USE NewStock INDEX NewStock
  PACK
ENDIF
CLOSE DATABASES
RETURN
```

4

الجزء الرابع

نظام حسابات العملاء

الفصل الثامن عشر

تصميـــم النظـــام

يتم فى هذا الجزء تصميم نظام حسابات العملاء (Accounts Receivable) الذى يشتمل على طباعة الفواتير آليا. ويتم فى هذا النظام تخزين إسم العميل (Customer) وعنوانه فى ملف قاعدة بيانات منفصل بالإضافة إلى الموازنة الحالية (Current balance) والموازنات السابقة (Aged balances). كما يتم تخزين بيانات السداد (Payments) فى ملف قاعدة بيانات منفصل. كما يتم تخزين بيانات الصرف (Charges) فى ملف قاعدة بيانات اخر ويقوم البرنامج بالربط بين هذه الملفات الثلاثة.

ودراسة هذا النظام تتيح لمخطط البرامج معرفة وسائل جديدة للتعامل مع الملفات المرتبطة (Relational Databases). كما يتم من خلالها دراسة استخدام ملفات الخطوات أو الإجراءات (Procedure Files) فى تسهيل كتابة البرامج وزيادة كفاءتها.

١٨ - ١ تعريف المشكلة

الهدف من نظام حسابات العملاء هو إنشاء قاعدة بيانات يتم من خلالها متابعة بيانات العملاء وتسجيل الموقف الشهرى (Monthly) لحركة السداد (Payments Transactions) وحركة الصرف (Charges Transactions) لكل حساب (Account) ويقوم النظام بإصدار فواتير (Invoices) لكل حساب توضح الموازنة الإبتدائية (Starting Balance) للشهر وكل حركة سداد أو صرف خلال هذا الشهر ثم الموازنة الحالية (Current Balance). كما يوفر النظام تقارير مختصرة (Summary Reports) للنشاط الشهرى لكل عميل. كما يوفر أيضا تقارير للموازنة خلال ٣٠ يوما و ٦٠ يوما و ٩٠ يوما وأكثر من ٩٠ يوما. كما يتيح النظام أيضا تخزين نسخة من فاتورة العميل حتى يستطيع العميل مراجعة أى بيانات سابقة.

ويقوم البرنامج كالعادة باستخدام القوائم الواضحة فى تسهيل تعامل المستخدم مع النظام.

١٨ - ٢ تحديد هيكل قاعدة البيانات

كما سبق الإيضاح فإن قاعدة البيانات تتكون من ثلاثة ملفات أولها ملف بيانات العميل (Customer) الذى يحتوى على كل البيانات الأساسية لهذا العميل والثانى ملف حركة الصرف (Charges) الذى يتضمن بيانات فواتير الصرف والأجزاء المصروفة وأسعارها والثالث ملف حركة السداد (Payments) الذى يتضمن بيانات الشيكات وموقف

تسديد الفواتير.

١٨ - ٢ - ١ ملف بيانات العميل (Customer.dbf)

يتكون ملف بيانات العميل من الحقول الموضحة بالشكل (١٨ - ١)

Field	Field Name	Type	Width	Dec
1	ACCOUNT_NO	Numeric	4	0
2	CUST_NAME	Character	25	
3	ADDRESS	Character	25	
4	PHONE	Character	13	
5	LAST_UPDAT	Date	8	
6	START_BAL	Numeric	8	2
7	CHG_CURR	Numeric	8	2
8	PAY_CURR	Numeric	8	2
9	BAL_30	Numeric	8	2
10	BAL_30	Numeric	8	2
11	BAL_90	Numeric	8	2
12	BAL_90 , PLUS	Numeric	8	2
13	TERMS	Character	20	

شكل (١٨ - ١)

والحقل رقم (١) يمثل رقم الحساب وهو حقل عددى. والحقل رقم (٢) يمثل إسم العميل وهو حقل حرفى يتكون من ٢٥ حرفا. والحقل رقم (٣) يمثل العنوان وهو حقل حرفى يتكون من ٢٥ حرفا. والحقل رقم (٤) هو حقل رقم التليفون وهو حقل حرفى يتكون من (١٣) حرفا. والحقل رقم (٥) هو حقل تاريخ آخر تعديل. والحقل رقم (٦) هو حقل الموازنة الابتدائية وهو حقل عددى مكون من ثمانية أرقام ورقمين عشريين. والحقل رقم (٧) هو حقل حركة الصرف الحالية (Current Charges). والحقل رقم (٨) هو حقل حركة السداد الحالية (Current Payment). والحقل رقم (٩) هو حقل الموازنة خلال شهر. والحقل رقم (١٠) هو حقل الموازنة خلال شهرين. والحقل رقم (١١) هو حقل الموازنة خلال ثلاثة شهور. والحقل رقم (١٢) هو حقل الموازنة خلال أربعة أشهر. والحقل رقم (١٣) هو حقل الدين (Credit).

ويستخدم حقل رقم الحساب (Account_No) فى ربط الملفات الثلاثة. ولذلك فمن الطبيعى أن تتم فهرسة الملفات الثلاثة بناء على هذا الحقل. ولفهرسة ملف العميل (Cnstomer.dbf) يتم كتابة السطر التالى من مشيرة النقطة .(Dot Prompt)

```
INDEX ON Account_No TO Cust_No
```

ولكن قد يحتاج المستخدم فى بعض الأحيان إلى البحث عن عميل معين بإسمه. لذلك يمكن إنشاء فهرس آخر بناء على حقل إسم العميل (Cust_name). ويمكن استخدام الدالة (UPPER) فى تحويل إسم العميل فى الفهرس إلى حروف كبيرة (Uppercase) لتسهيل البحث عن إسم العميل سواء أدخل المستخدم الإسم بحروف صغيرة أو حروف كبيرة. ولإنشاء هذا الفهرس يتم كتابة السطر التالى من مشيرة النقطة (Dot Prompt).

```
INDEX ON UPPER(Cust_name) TO Cust_Name
```

١٨ - ٢ - ٢ ملف حركة الصرف (Charges.dbf)

يتكون ملف حركة الصرف (Charges.dbf) من الحقول الموضحة فى الشكل (١٨ - ٢). ويلاحظ أن حقل رقم الحساب (Account_no) موجود فى الملف بنفس الإسم ونفس النوع (Type) ونفس العرض (Width). وذلك لأنه الحقل الذى سوف يستخدم كمفتاح (Key Field) لربط الملفات الثلاثة.

ويتم إنشاء فهرس لهذا الملف بناء على حقل رقم الحساب. وذلك بكتابة السطر التالى من مشيرة النقطة (Dot Prompt).

```
INDEX ON Account_No TO ChrgNo
```

Field	Field Name	Type	Width	Dec
1	ACCOUNT_NO	Numeric	4	0
2	INVOICE_NO	Numeric	6	0
3	PART_NO	Character	5	
4	QTY	Numeric	7	2
5	UNIT_PRICE	Numeric	9	2
6	AMOUNT	Numeric	9	2
7	DATE	Date	8	
8	DESCRIPT	Character	20	
9	BILLED	Logical	1	

شكل (١٨ - ٢)

والحقل رقم (١) يمثل رقم الحساب ، والحقل رقم (٢) يمثل رقم الفاتورة، والحقل رقم (٣) يمثل حقل رقم الجزء ، والحقل رقم (٤) يمثل حقل الكمية ، والحقل رقم (٥) يمثل حقل سعر الوحدة. والحقل رقم (٦) يمثل حقل الكمية الكلية المصروفة ، والحقل رقم (٧) يمثل حقل تاريخ الشراء ، والحقل رقم (٨) يمثل حقل وصف عملية الشراء ، والحقل رقم (٩) هو حقل منطقى يوضح إذا كان الحساب قد تم تسديده أم لا.

١٨ - ٢ - ٣ ملف السداد (Payments.dbf)

أما ملف السداد (Payments.dbf) فيتم تكوينه من الحقول الموضحة فى الشكل (١٨ - ٣).

والحقل رقم (١) يمثل رقم الحساب ، والحقل رقم (٢) يمثل رقم الشيك ، والحقل رقم (٣) يمثل حقل الكمية الكلية المدفوعة ، والحقل رقم (٤) يمثل حقل تاريخ الشراء ، والحقل رقم (٥) يمثل حقل وصف العملية، والحقل رقم (٦) هو حقل منطقى يوضع إذا كان الدفع قد تم تسجيله أم لا.

Field	Field Name	Type	Width	Dec
1	ACCOUNT_NO	Numeric	4	0
2	CHECK_NO	Character	5	
3	AMOUNT	Numeric	9	2
4	DATE	Date	8	
5	DESCRIPT	Character	30	
6	POSTED	Logical	1	

شكل (١٨ - ٣)

ويتم إنشاء ملف الفهرس لهذا الملف بكتابة السطر التالى من مشيرة النقطة (Dot Prompt) كالآتى :

INDEX ON Cust_no TO PayNo

١٨ - ٣ حفظ البيانات التاريخية

يقوم نظام حسابات العملاء، كما سبق الإيضاح على المتابعة الشهرية أى تسجيل حركة الصرف والتوريد خلال الشهر. وفى نهاية الشهر يتم تجميع الفواتير وإرسالها إلى العملاء.

ولكن ماذا لو أراد المستخدم الرجوع إلى بيانات سابقة عن حسابات تم تسديدها ؟

هناك ثلاثة إحتمالات للتعامل مع البيانات القديمة وهى كالآتى :

١- مسح كل حركة للصرف أو السداد بمجرد تسديدها. وتمتاز هذه الطريقة بتوفير مساحة تخزينية على القرص ولكن يعيبها عدم القدرة على الرجوع إلى أى بيانات سابقة عن أى حركة تم تسديدها.

٢- ترك كل حركة صرف أو سداد مخزنة فى الملف حتى بعد تسديدها مع تمييز الحركة التى تم تسديدها. وفائدة هذه الطريقة أنها تحتفظ بالبيانات السابقة عن أى حركة مهما كان تاريخها. ولكن يعيبها أنها سوف تؤدى إلى تضخم

ملفى الصرف (Charges) والسداد (Payment) إلى حجم لانهائى.

٣ - نقل كل حركة صرف أو سداد بمجرد تسديدها من ملفى الصرف والسداد إلى ملف تاريخى (Historical file). وفائدة هذه الطريقة أنها تؤدى إلى الإحتفاظ بملفى الصرف والتوريد صغيرين نسبيا مع الإحتفاظ بجميع البيانات السابقة عن أى حركة سبق تسديدها.

ومن خلال نظام حسابات العملاء الجارى شرحه سيتم استخدام الإحتمال الثالث وهو نقل جميع السجلات التى تم تسديدها من ملفى الصرف والسداد إلى ملف تاريخى (Historical File) ويمكن تخزين هذا الملف التاريخى فى قرص أو أقراص منفصلة. ولإنشاء ملف تاريخى للملف الصرف (Charges.dbf) يتم نسخ هيكل الملف (Structure) إلى ملف جديد يتم تسميته (BillHist.dbf) مثلا. وذلك بكتابة السطرين التاليين من مشيرة النقطة (Dot Prompt).

```
USE Charges
COPY STRUCTURE TO BillHist
```

ولإنشاء ملف تاريخى مماثل بالنسبة للملف السداد (Payments) يتم كتابة السطرين التاليين من مشيرة النقطة.

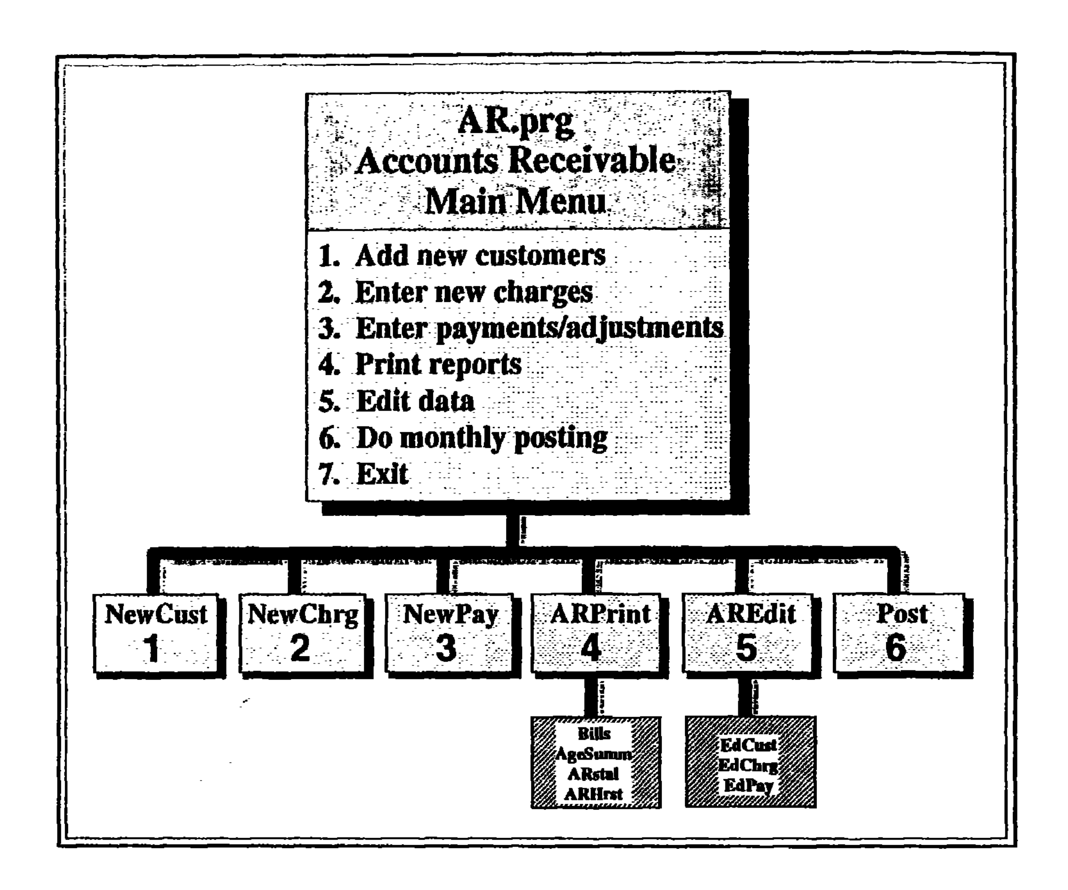
```
USE Payments
COPY STRUCTURE TO PayHist
```

وفى الأجزاء التالية سيتم إيضاح كيفية نقل السجلات التى تصبح غير مطلوبة إلى الملفات التاريخية من خلال برنامج حسابات العملاء.

١٨ - ٤ تركيب البرنامج

يتكون برنامج حسابات العملاء من عدة برامج منفصلة يقوم أحدها بإضافة عملاء جدد، والثانى بطباعة التقارير والفواتير والثالث بتعديل بيانات أحد العملاء والرابع بالترحيل الشهرى و ... الخ.

والشكل الهرمي (Hierarchical) للبرنامج يتضح من الشكل (١٨ - ٤). ويلاحظ من الشكل أن البرنامج أكثر تعقيدا من برنامج المخازن حيث أنه يشمل عدة ملفات. ولذلك سيتم فى الأجزاء التالية استخدام وسيلة جديدة لزيادة سرعة تنفيذ البرنامج وتقليل عدد الملفات المستخدمة. وهذه الوسيلة هى استخدام ملف الخطوات (Procedure File) وسيتم شرحها فى الفصل التالى.



شكل (١٨ - ٤)

الفصل التاسع عشر

ملفـــات الخطـوات

فى هذا الفصل سيتم شرح إحدى الوسائل المتقدمة لكتابة البرامج بواسطة (DBase III+) أو برامج عائلة (DBase) الأخرى وهى استخدام ملفات الخطوات (Procedure Files) والمعاملات (Parameters). وسوف يتم استخدام هذه الوسيلة فى برنامج حسابات العملاء (A/R).

وملف الخطوات هو ملف يحتوى على برامج صغيرة (Routines) ويمكن استخدام كل برنامج فى عدة أماكن من البرنامج الرئيسى باستدعائه عند الحاجة إليه أى أنه يعتبر مكتبة برامج (Library). ويمكن إنشاء عدة ملفات خطوات ولكن لايمكن فتح أكثر من ملف فى نفس الوقت. وهى تؤدى إلى تسهيل كتابة البرامج وسرعة تنفيذها كما أنها تتغلب على مشكلة زيادة عدد الملفات المفتوحة التى تظهر فى البرامج الكبيرة. وذلك لأن ملف الخطوات يعتبر ملفا واحدا رغم أنه يحتوى على عدة برامج.

إرجع إلى الأمر (SET PROCEDURE TO) فى الكتاب رقم (٥) من مجموعة كتب دلتا.

١٩ - ١ إستخدام ملف الخطوات فى برنامج حسابات العملاء

يستخدم مع برنامج حسابات العملاء ملف خطوات واحد يحتوى على ثلاثة برامج. وهذا الملف عند فتحه باستخدام الأمر (SET PROCEDURE TO) يتم تحميله فى الذاكرة المؤقتة (RAM). وهذا يؤدى إلى سرعة تشغيل البرامج داخله حيث لاتصبح هناك حاجة دائما إلى نقل هذه البرامج من القرص إلى الذاكرة.

وكل برنامج (Procedure) موجود داخل الملف يمكن تحويله إلى برنامج عام (General) يمكن استخدامه فى أى برامج أخرى. ويتم ذلك باستخدام الأمر (PARAMETERS). وهذا الأمر يؤدى إلى إدخال معاملات معينة إلى البرنامج ليناسب تطبيقا محددا. ويمكن استخدام معاملات أخرى لتنفيذ البرنامج فى تطبيقات أخرى متعددة. وهذا سيتم إيضاحه عند دراسة البرامج الثلاثة (Procedures) المستخدمة مع برنامج حسابات العملاء.

١٩ - ٢ برنامج العنوان (Title)

يستخدم هذا البرنامج لكتابة أى عنوان وكتابة تاريخ اليوم الحالى مع رسم خط عرضى بعرض الشاشة. ويمكن استخدام هذا البرنامج مع أى شاشة يتم تصميمها للمستخدم.

ويبدأ هذا البرنامج بالأمر (PROCEDURE) يليه إسم البرنامج (Title) وذلك فى السطر الأول. وفى السطر الثانى يتم كتابة الأمر (PARAMETERS) يليه إسم المعامل أو المعاملات التى سوف يتم استخدامها عند استدعاء البرنامج. والسطر التالى يقوم بمسح الشاشة. والسطر التالى يقوم بكتابة العنوان (Title). والسطر التالى يقوم بكتابة تاريخ اليوم الحالى والوقت الحالى. والسطر التالى يقوم برسم خط عرضى بعرض الشاشة وتخزينه فى المتغير (Uline) مع ملاحظة أن المتغير (Uline) يكون قد سبق تعريفه فى البرنامج الرئيسى كما تم فى برنامج المخازن. وسطور البرنامج الذى سبق شرحه تكون كالآتى :

```
PROCEDURE Title
PARAMETERS Title
   * - - - - -  Display screen title
   CLEAR
   @ 2,1  SAY Title
   @ 2,60 SAY DTOC(DATE()) + "  " + TIME()
   @ 3,0  SAY Uline
   ?
   ?
 RETURN
```

والشىء المثير فى هذه العملية أنه فى أى نقطة من البرنامج الرئيسى عندما يراد عرض شاشة للمستخدم متضمنة العنوان وتاريخ اليوم الحالى والوقت والخط العرضى فإنه يكفى كتابة الأمر (DO) يليه إسم هذا البرنامج ثم العنوان المراد كتابته. وذلك كالآتى مثلا :

```
Do Title WITH "Accounts Receivable Main Menu"
```

وكلمة (WITH) تنبه برنامج (+DBase III) أن البيانات التالية لها هى المعاملات التى سبق تعريفها فى البرنامج الفرعى (Procedure). ولذلك فعند تنفيذ هذا البرنامج الفرعى ووصول البرنامج إلى الأمر (PARAMETERS) فإنه يستبدل العنوان (Title) بالعنوان الذى تم إدخاله وهو ("Account Receivable Main Menu"). لذلك فعند تنفيذ البرنامج يظهر الآتى على الشاشة.

Accounts Receivable Main Menu   02/20/90   12:50:30

---

١٩ - ٣ برنامج رسائل الأخطاء

من البرامج التى تستخدم أيضا فى ملفات الخطوات (Procedure Files) برنامج رسائل الأخطاء. وهذا البرنامج يستخدم فى عرض رسائل الأخطاء على الشاشة لتوضح للمستخدم الخطأ الذى وقع فيه والعمل الذى يجب تنفيذه للتغلب على هذا الخطأ. وهذا البرنامج يتكون من السطور التالية :

```
PROCEDURE Error
PARAMETERS Message
   * - - - - - Display error message.
   @ 20,0 CLEAR
   @ 20,3 SAY Message
   ? CHR(7)
   WAIT "Press any key to try again"
RETURN
```

وهذا البرنامج عند استدعائه بواسطة الأمر (DO) يقوم بمسح الشاشة إبتداء من السطر (20) ولأسفل وذلك لكى تظل البيانات موجودة على الشاشة. ثم يقوم بعرض رسالة الخطأ (Message) التى يتم إدخالها مع الأمر (DO) كما يشغل الجرس (Bell) وينتظر حتى يضغط المستخدم على أى مفتاح أو ينفذ أى شىء يكون موضحا فى الرسالة (Message). فمثلا عند كتابة الأمر التالى :

```
DO Error WITH "No such customer number!"
```

فإن البرنامج يعرض الآتى أسفل الشاشة :

```
No such customer number !
Press any key to try again
```

كما يشغل الجرس (Bell) وينتظر من المستخدم الضغط على أى مفتاح لاستكمال تنفيذ البرنامج الرئيسى.

١٩ - ٤ برنامج التحقق من رقم العميل

هذا هو البرنامج الثالث المستخدم بواسطة برنامج حسابات العملاء وهو يتيح للمستخدم استعمال إسم العميل أو رقمه فى البحث عن بيانات عميل محدد. ويتم ذلك من خلال أمر واحد كالآتى :

```
DO GetCust WITH M_Cust_No,M_Cust_Name,M_Address,Exiting
```

وهذا الأمر يؤدى إلى تنفيذ الآتى :

١- عرض شاشة للمستخدم لعرض إسم العميل أو رقمه.
٢- إذا أراد المستخدم الخروج يتم إخبار البرنامج القائم بالإستدعاء بذلك.
٣- البحث عن أرقام الحساب الخاصة بالإسم الذى تم إدخاله وعرضها على الشاشة.
٤- فى حالة إدخال رقم حساب غير موجود يتم عرض رسالة خطأ والسماح للمستخدم بالمحاولة مرة ثانية.
٥- العودة للبرنامج القائم بالإستدعاء مع إعطائه رقم العميل وإسمه وعنوانه وعرض هذه البيانات على الشاشة لكى يتحقق المستخدم أن هذا هو العميل المطلوب.

وهذا البرنامج سوف يستخدم كثيرا فى برنامج حسابات العملاء ولذلك فسوف يتم شرحه بالتفصيل فى نهاية هذا الفصل.

١٩ - ٥ إنشـــاء ملف الخطوات

يتم إنشاء ملف الخطوات بنفس طريقة إنشاء أى ملف أوامر (Command file). وذلك باستخدام الأمر (MODIFY COMMAND) ثم كتابة إسم الملف ونسميه فى هذه الحالة (Proclib1.dbf) وهو إختصار (Procedure Library 1). وملف الخطوات يمكن أن يحتوى على عدد من البرامج بحد أقصى ٣٢ برنامجا. ولكن فى نظام حسابات العملاء فإن الملف (Proclib1) يحتوى على ثلاثة برامج فقط كما سبق الإيضاح.

وكل برنامج فرعى (Procedure) يبدأ بالأمر (PROCEDURE) يليه إسم البرنامج وينتهى بالأمر (RETURN). وفى حالة استخدام المعاملات (Parameters) يتم استخدام الأمر (PARAMETERS) فى السطر التالى مباشرة لسطر الأمر (PROCEDURE).

والسطور التالية توضح أوامر الملف (ProcLib1).

```
* * * * * * * * * * * * * * * Proclib1.prg
*    General Procedure for the AR System

* - - - - - Display screen title.
PROCEDURE Title
PARAMETERS Title
   CLEAR
   @ 1,0  SAY Title
   @ 1,60 SAY DTOC(DATE()) + " " + TIME()
   @ 2,0  SAY Uline
   ?
   ?
RETURN

* - - - - - Display error message.
PROCEDURE Error
PARAMETERS Message
```

```
   @ 20,0
   CLEAR
   @ 20,3 SAY Message
   ? CHR(7)
   WAIT "Press any key to try again"
RETURN

* - - - - - Look up customer by number or name
PROCEDURE GetCust
PARAMETERS M_Cust_No, M_Name, M_Address , Exiting

   * - - Set up loop for validating customer number.
   * - - Enter customer number or customer name and
   * - - look for it.
   Valid = .F.
   DO WHILE .NOT. Valid

      * - - - Get customer name or number
      Lookup = SPACE(20)
      @ 4,0  CLEAR
      @ 15,5 SAY "Enter customer number or name" GET Lookup
      @ 17,5 SAY "Press Return to exit"
      READ

      * - - - If nothing entered , return "exiting"
      IF Lookup = "  "
                  Exiting = .T.
                  RETURN
      ENDIF (Lookup is blank)

      * Lookup by name if name entered.
      IF ASC(Look up) > = 65
```

```
            Lookup = UPPER(TRIM(Lookup)
            SET INDEX TO CustName, CustNo
            SET EXACT OFF
            SEEK Lookup
            * - - - - - If name found
            IF FOUND()

                * - - - -Display customers with required
                * - - - - - name
                M_Cust_No = Cust_No
                @ 5,0  CLEAR
                @ 6,0  SAY "Number   Name   Address"
                ?
                DISPLAY OFF WHILE UPPER(Cust_Name) = ;
                    Lookup Cust_No , Cust_Name, Address
                @ 22,2 SAY "Enter customer number" GET ;
                    M_Cust_No PICTURE "9999"
                READ
                Lookup = STR(M_Cust_No , 4)
                ELSE(If name not found)
                DO Error WITH "Not found"
            ENDIF(name not found)
ENDIF(name entered)

* - - - - - Lookup by customer number.
IF VAL(Lookup) > 0
        M_Cust_No = Val(Lookup)
        SET INDEX TO CustNo, CustName
        SEEK M_Cust_No
        * - - - - - If found, Continue, else ask again.
        IF FOUND()
            Valid = .T.
            M_Name = TRIM(Cust_Name)
```

```
                    M_Address = TRIM(Address)
               ELSE
                    DO Error with "Not found"
               ENDIF
          ENDIF (number entered)
ENDDO (while invalid entry)
RETURN
```

١٩ - ٦ فتح ملف الخطوات

حتى يمكن تشغيل أى برنامج داخل ملف الخطوات يلزم فتح هذا الملف. ويتم ذلك باستخدام الأمر (SET PROCEDURE TO) ثم كتابة إسم الملف المطلوب. فمثلا لفتح ملف الخطوات الخاص ببرنامج حسابات العملاء (A/R) يتم كتابة السطر التالى :

```
SET PROCEDURE TO ProcLib1
```

وتنفيذ هذا الأمر يؤدى إلى تحميل الملف (ProcLib1) فى الذاكرة المؤقتة (RAM) وبالتالى يمكن تنفيذ أى برنامج داخل هذا الملف باستخدام الأمر (DO) كما سبق الإيضاح.

ولإغلاق ملف الخطوات المفتوح يمكن كتابة الأمر (CLOSE PROCEDURE): كما يمكن فتح ملف خطوات آخر حيث أن هذا يؤدى إلى إغلاق ملف الخطوات السابق مع ملاحظة أنه لايمكن فتح أكثر من ملف خطوات فى نفس الوقت.

١٩ - ٧ إدخال المعاملات (Parameters)

يستخدم الأمر (PARAMETERS) فى إدخال المعاملات إلى برامج الخطوات (Procedures) كما سبق الإيضاح. وهو يؤدى إلى إدخال المعاملات التى يتم كتابتها مع الأمر (DO) فى المتغيرات الموجودة بعده. ويتم إدخال المعاملات بنفس الترتيب ويجب ملاحظة أن عدد المعاملات التى يتم إدخالها بعد الأمر (DO) يجب أن يكون مطابقا لعدد المتغيرات التى يتم كتابتها بعد الأمر (PARAMETERS) وإلا فإن البرنامج يتوقف ويعطى رسالة خطأ.

فمثلا السطور التالية تمثل إنشاء برنامج خطوات إسمه (GetArea) يستخدم المتغيرات الثلاثة التالية (Length) ، (Width) ، (Area).

```
PROCEDURE GetArea
PARAMETERS Length, Width, Area
   Area = Length * Width
RETURN
```

ولتشغيل هذا البرنامج مع إدخال الطول (20) والعرض (25) والمساحة (Area) يتم كتابة الآتى :

```
Area = 0
DO GetArea WITH 20, 25, Area
```

وعند السؤال عن المساحة (Area) يتم كتابة الآتى :

```
? Area
```

فيلاحظ ظهور العدد (500).

ويلاحظ هنا ضرورة تعريف المتغير (Area) قبل استخدامه. ويمكن استخدام أى متغيرات (X, Y, Z) كمعاملات مع تعريفها قبل استخدامها وذلك كالآتى :

```
X = 5
Y = 10
Z = 0
DO GetArea WITH X , Y, Z
```

وعند السؤال عن قيمة المتغير (Z) كالآتى :

```
? Z
```

يلاحظ ظهور العدد (50).

ويجب ملاحظة أن الأمر (PARAMETERS) لايتم استخدامه فى إدخال المعاملات إلى البرنامج فقط ولكنه يستخدم فى إخراج قيم من البرنامج مثل المعامل (Area) مثلا. كما يجب ملاحظة أن إدخال المعاملات لايستخدم فقط مع برامج الخطوات(Procedures) ولكنه يستخدم أيضا مع أى برنامج يتم إنشاؤه بواسطة برامج عائلة (DBase).

١٩ - ٨ دراسة برنامج الخطوات (GetCust)

هذا البرنامج مكتوب فى ملف الخطوات (ProcLib1) السابق شرحه. والبرنامج يبدأ بالسطرين التاليين :

```
PROCEDURE GetCust
PARAMETERS M_Cust_No, M_Name, M_Address , Exiting
```

والمعاملات (Parameters) فى هذا البرنامج تستخدم أساسا لإعادة البيانات المطلوبة إلى البرنامج القائم بالإستدعاء.

والبرنامج يقوم بسؤال المستخدم عن الإسم أو الرقم المطلوب البحث عنه ويقوم بتخزين هذا الإسم أو الرقم فى المتغير (Lookup). وإذا لم يدخل المستخدم أى إسم فإن برنامج الخطوات يعود إلى البرنامج المستدعى (Calling Program) مع إعادة القيمة (.T.) فى المتغير (Exiting). ويتم ذلك من خلال السطور التالية :

```
IF Lookup = "  "
   Exiting = .T.
   RETURN
ENDIF(lookup is blank)
```

وإذا أدخل المستخدم إسم عميل معين فإن البرنامج يحول هذا الإسم إلى حروف كبيرة (Uppercase) ثم يبحث عن هذا الإسم. وذلك من خلال السطور التالية :

```
IF ASC(look up) > = 65
```

```
Lookup = UPPER(TRIM(Lookup))
SET INDEX TO CustName, CustNo
SET EXACT OFF
SEEK Lookup
```

ويتم استخدام الدالة (ASC) لمعرفة ما إذا كان المستخدم أدخل إسما أم رقما. حيث أن كود الآسكى للحرف (A) هو (65). أما الأرقام فإن الكود الخاص بها يكون أقل من (65). ولذلك فإن تحقق الشرط بعد (IF) يعنى أن المستخدم يدخل حروفا وليس أرقاما. وهذا يعنى أنه أدخل الإسم وليس رقم العميل. وفى هذه الحالة يتم فتح ملف الفهرس (CustName) باعتباره الفهرس الرئيسى ثم يتم البحث عن هذا الإسم. ويلاحظ هنا استخدام الأمر (SET EXACT OFF) للسماح للمستخدم بإدخال أى عدد من الحروف والحصول على كل الأسماء التى تبدأ بهذه الحروف.

وعند العثور على هذا الإسم يتم عرض بيانات جميع الأسماء المشتركة فى هذا الإسم حتى يختار منهم المستخدم الإسم المطلوب. وفى هذه الحالة يقوم بإدخال رقم الحساب الخاص به. وهذا يؤدى إلى الوصول إلى السجل المطلوب.

ويتم ذلك من خلال السطور التالية :

```
IF FOUND()
   *----- Display customers with required
   *----- name
   M_Cust_No = Cust_No
   @ 5,0  CLEAR
   @ 6,0  SAY "Number   Name   Address"
   ?
   DISPLAY OFF WHILE UPPER(Cust_Name) = ;
     Lookup Cust_No , Cust_Name, Address
   @ 22,2 SAY "Enter customer number" GET ;
     M_Cust_No PICTURE "9999"
   READ
   Lookup = STR(M_Cust_No , 4)
```

وفى حالة عدم العثور على هذا الإسم يتم عرض رسالة خطأ. ويلاحظ هنا أنه تم استخدام برنامج الخطوات (Error.prg) الموجود فى نفس الملف. وهذا يعنى أنه يمكن إستدعاء برنامج خطوات من برنامج خطوات آخر موجود داخل نفس ملف الخطوات (Procedure File).

والسطور التالية توضح هذه العملية.

```
   ELSE(If name not found)
      DO Error WITH "Not found"
   ENDIF(name not found)
ENDIF(name entered)
```

والجزء التالى من البرنامج يعالج حالة إدخال المستخدم لرقم العميل دون إسمه. وحيث أن القيم التى تنتج من الدالة (VAL) مع أى سلسلة حرفية تكون صفرا. فإن الشرط الموجود بعد (IF) لايتحقق إلا إذا كان ما يدخله المستخدم فى المتغير (Lookup) أعدادا وليس حروفا.

وحيث أن الحقل (Cust_No) هو حقل عددى لذلك يجب عند البحث عنه التأكد من أن القيمة الجارى مقارنتها قيمة عددية. ولذلك تستخدم الدالة (VAL) لتحويل المتغير (Lookup) إلى متغير عددى. ثم يتم البحث عن هذا المتغير بعد فتح الفهرس الخاص برقم العميل (CustNo). ويتم ذلك من خلال السطور التالية :

```
IF VAL(Lookup) > 0
   M_Cust_No = Val(Lookup)
   SET INDEX TO CustNo, CustName
   SEEK M_Cust_No
```

وإذا تم العثور على رقم العميل يتم تغيير المتغير (Valid) إلى (.T.) وهذا يؤدى إلى عدم تنفيذ الحلقة التكرارية مرة ثانية. ثم يتم تخزين الإسم (Cust_Name) الخاص بالسجل الذى تم الوصول إليه فى المتغير (M_Name) والعنوان الخاص به (Address) فى المتغير (M_Address). وإذا لم يتم العثور على رقم العميل يتم عرض رسالة خطأ (Error Message). والسطور التالية توضح هذه العملية :

```
   IF FOUND()
        Valid = .T.
        M_Name = TRIM(Cust_Name)
        M_Address = TRIM(Address)
   ELSE
        DO Error with "Not found"
   ENDIF(not eof)
ENDIF(number entered)
ENDDO(While invalid entry)
RETURN
```

ويلاحظ خلال هذا البرنامج أن ملف قاعدة البيانات لم يتم فتحه وذلك لأنه يكون قد تم فتحه من خلال البرنامج الرئيسى لنظام حسابات العملاء. كما يجب ملاحظة أن ملف الخطوات يجب فتحه من خلال البرنامج الرئيسى حتى يمكن استخدام البرامج الموجودة فيه فى البرامج الفرعية للنظام.

الفصل العشرون

برامج القائمة الرئيسية والادخال والتعديل

فى هذا الفصل سوف يتم دراسة برنامج القائمة الرئيسية لحسابات العملاء (Main Menu) وبرنامج إدخال عملاء جدد وبرنامج إدخال حركة الصرف وبرنامج إدخال حركة السداد بالإضافة إلى برامج التعديل. وحيث أن هذه البرامج لاتختلف عن برامج المخازن السابق شرحها من حيث تصميمها المنطقى لذلك فسوف يتم كتابتها دون شرحها مع شرح أى أوامر جديدة لم يسبق استخدامها.

٢٠ - ١ برنامج القائمة الرئيسية

هذا البرنامج لايختلف عن باقى برامج القائمة الرئيسية حيث أنه يعرض قائمة على المستخدم للاختيار منها. وبناء على اختيار المستخدم يقوم البرنامج بالتفرع إلى البرامج الأخرى. ولكن فى هذا البرنامج يتم كتابة الأمر التالى :

```
SET PROCEDURE TO ProcLib1
```

وذلك لفتح ملف الخطوات (Proclib1) السابق شرحه فى الفصل السابق. كما يجب كتابة الأمر (CLOSE PROCEDURE) قبل نهاية البرنامج.

وعند تشغيل هذا البرنامج تظهر الشاشة الموضحة بالشكل (٢٠ - ١)

```
02/20/90    08:30:45
Accounts Receivable Edit Menu

      1. Add new customers
      2. Enter new charges
      3. Enter payments/Adjustments
      4. Print data
      5. Edit data
      6. Do monthly posting
      7. Exit

Enter choice (1 - 7)
```

شكل (٢٠ - ١)

• ويتكون هذا البرنامج من السطور التالية :

```
* * * * * * * * * * * * * * * * * * * * * * *AR.prg
*   Accounts Receivable Main menu
CLEAR ALL

* - - - - Declare variables as public for passing to
* - - - - procedures.
PUBLIC M_Cust_No, M_Name, M_Address, Message, Exiting

* - - - - - - Open Procedure file ProcLib1.prg
SET PROCEDURE TO ProcLib1

* - - - - - - If color monitor in use , set colors .
IF ISCOLOR()
        SET COLOR TO GR+/B , W+/RB
ENDIF

* - - - - - - Set Parameters
SET BELL OFF
SET DELETED ON
SET DEVICE TO SCREEN
SET HEADING OFF
SET STATUS OFF
SET TALK OFF

* - - - - Get the date (RUN needs abount 320KB RAM).
?
RUN DATE

* - - - - - - Create underline variable Uline.
Uline = REPLICATE("_" , 80)
```

```
* - - - - - - Set up a loop for the main menu.
Choice = 0
DO WHILE Choice # 7

        CLEAR
        * - - - - - - Print screen title
        DO Title WITH "Accounts Receivable Main Menu"
        TEXT
                1.  Add new customers
                2.  Enter new charges
                3.  Enter payments / adjustments
                4.  Print reports
                5.  Edit data
                6.  Do monthly posting
                7.  Exit
        ENDTEXT
        @ 23,1 SAY "Enter choice (1 - 7) " ;
            GET Choice PICT "9" RANGE 1,7
        READ
        * - - - - - Branch accordingly.
        DO CASE
                CASE Choice = 1
                                DO NewCust
                CASE Choice = 2
                                DO NewChrg
                CASE Choice = 3
                                DO NewPay
                CASE Choice = 4
                                DO ARPrint
                CASE Choice = 5
                                DO AREdit
                CASE Choice = 6
```

```
                DO Post
        ENDCASE
ENDDO(while choice # 7)

* - - - - -  Close procedure file and exit.
CLOSE PROCEDURE
CLEAR
QUIT
```

ويلاحظ فى هذا البرنامج استخدام الأمر(PUBLIC) مع المتغيرات(M_Cust_No)، (M_Name) ، (M_Address) ، (Exiting) وذلك حتى يمكن استخدام هذه المتغيرات فى جميع البرامج الفرعية دون الحاجة إلى إعادة إنشائها من جديد. كما أن هذا يعتبر ضروريا لاستخدام برنامج الخطوات (GetCust). وحتى لا نحتاج إلى إعادة إنشاء هذه المتغيرات فى كل مرة يراد فيها استخدام هذا البرنامج.

ويلاحظ أيضا استخدام برنامج الخطوات (Title) لكتابة عنوان الشاشة ورسم خط فيها.

ويلاحظ أيضا استخدام الأمر (SET COLOR) فى تجميل الشاشة وزيادة جاذبيتها. وهذا الأمر مع المعاملات الموجودة معه يؤدى إلى تكوين خلفية زرقاء مع الكتابة باللون الأصفر فى الشاشة الرئيسية. كما يؤدى إلى تكوين خلفية بنفسجية مع الكتابة باللون الأبيض فى الأعمدة الضوئية (Highlights) ويمكن تعديل هذه الألوان حسب الحاجة.

والأمر (RUN DATE) تم استخدامه لتشغيل برنامج التاريخ (DATE) الموجود فى نظام التشغيل (MS_DOS) حتى يسمح للمستخدم بتعديل التاريخ الحالى حسب الحاجة. ويجب ملاحظة أن استخدام الأمر (RUN) من خلال (+DBase III) يتطلب ذاكرة مؤقتة لاتقل عن ٣٢٠ ك بايت.

٢٠ - ٢ برنامج إضافة العملاء (NewCust.prg)

هذا البرنامج لايختلف كثيرا عن برنامج إضافة أرقام الأجزاء الجديدة فى نظام المخازن السابق شرحه ولكن نظرا لأن نظام حسابات العملاء يعتمد على ربط الملفات لذلك فإن هذا

البرنامج يضيف جزءا جديدا لاختبار كل رقم عميل جديد يتم إدخاله والتأكد من عدم تكراره. كما أن البرنامج يقوم بزيادة رقم العميل آليا مع كل إضافة لعميل جديد ويسمح للمستخدم بقبول هذا الرقم أو إدخال رقم آخر جديد.

وعندما يختار المستخدم الرقم (1) من القائمة الرئيسية للبرنامج الرئيسى (AR.prg) فإن البرنامج يمسح الشاشة ويعرض الرسالة التالية :

Enter Customer number (0 to quit) : 1001

فى هذه الحالة يستطيع المستخدم الضغط على مفتاح الإدخال للموافقة على هذا الرقم أو كتابة صفر (0) للرجوع إلى القائمة الرئيسية أو كتابة رقم حساب آخر.

وعندما يكتب المستخدم رقما موجودا تظهر الرسالة التالية :

Number already in use!

ثم يسمح البرنامج للمستخدم بالمحاولة مرة ثانية.

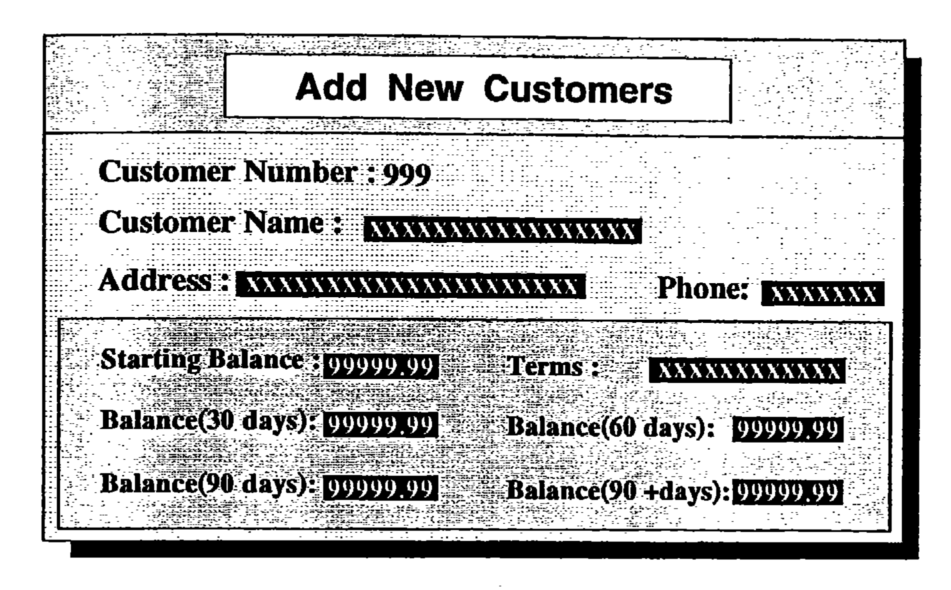
وعندما يكتب المستخدم رقما غير موجود تظهر شاشة الإدخال حتى يستطيع المستخدم إدخال باقى بيانات هذا العميل. والشكل (٢٠ - ٢) يوضح شاشة الإدخال المستخدمة فى هذا البرنامج. والتى يتم إنشاؤها من خلال قوائم برنامج المساعد (Assistant) أو عن طريق كتابة السطرين التاليين من مشيرة النقطة (Dot Prompt).

```
USE Customer
CREATE SCREEN FNewCust
```

وفى هذه الحالة تظهر قوائم الإختيارات التى يتم عن طريقها تحميل الحقول المطلوب ظهورها على السبورة (Blackboard). ثم يتم تحديد أماكن هذه الحقول على الشاشة كما سبق الإيضاح فى الكتاب الخامس من مجموعة كتب دلتا.

وحيث أن رقم العميل يتم التحكم فيه من خلال البرنامج لذلك يجب عدم السماح للمستخدم بتعديله. وهذا يتم تنفيذه عند تصميم الشاشة حيث يتم تحريك مؤشر التصحيح إلى العمود الضوئى المقابل لرقم العميل (Customer Number) ثم الضغط على مفتاح

(F10) ثم الضغط على مفتاح الإدخال عند وقوف المؤشر على الاختيار (ACTION) لتحويله إلى (Display/Say). ثم يتم الضغط على مفتاح (F10) مرة أخرى للرجوع إلى السبورة (Blackboard). ويلاحظ فى هذه الحالة إختفاء العمود الضوئى (Highlight) مع بقاء الأرقام (9999) الدالة على نوع الحقل. وهذا يعنى أن المستخدم يرى رقم العميل ولكن لايمكنه تعديله.



شكل (٢٠ - ٢)

والبرنامج المستخدم فى إضافة العملاء يسمى (NewCust.prg) ويتكون من السطور التالية :

```
* * * * * * * * * * * * * * * * NewCust.prg
*    Add new customers to the A/R system.
*    Called from AR main menu
USE Customer INDEX CustNo, CustName

* - - Store largest customer number to memory variable
* - - Next_No.
GO BOTT
Next_No = Cust_No
```

```
*----- Print screen title.
DO Title WITH "Add New Customers"

*----- Set up loop for adding customers.
Exiting = .F.
DO WHILE .NOT. Exiting
    *----- Increment customer number by 1
    Next_No = Next_No + 1
    *--- Suggest next number , but allow user to change it.
    @ 15,5 SAY "Enter customer number (0 to quit): " ;
       GET Next_No PICT "9999"
    @ 17,5 SAY "Press Return to accept number "
    READ
    *---- If zero entered , return to main menu.
    IF Next_No = 0
        Exiting = .T.
        LOOP
    ENDIF

    *--- Check to see if number is already taken.
    *--- If it is,loop and ask for another number.
    SEEK Next_No
    IF FOUND()
        ? "Number already in use" , CHR(7)
        LOOP
    ENDIF

    *--If next number isn't taken , add new record.
    *--using FNewCust format screen.
    APPEND BLANK

    REPLACE Cust_No WITH Next_No
```

```
        REPLACE Term WITH "Net30"
        REPLACE Last_Updat WITH DATE()
        SET FORMAT To FNewCust
        READ
        SET FORMAT TO
        @ 4,0  CLEAR
ENDDO (While not exiting)
* - - - - -  End of program
CLOSE DATABASES
RETURN
```

ويلاحظ فى هذا البرنامج استخدام المتغير (Next_No) فى زيادة رقم السجل بواحد آليا وكذلك فى اختبار رقم العميل والتأكد من عدم وجوده فى ملف العملاء (Customer.dbf).

٢٠ - ٣ برنامج إضافة حركة الصرف (Newchrg.prg)

يتم تشغيل هذا البرنامج عندما يختار المستخدم الرقم (2) من القائمة الرئيسية للبرنامج (AR.prg).

ويستخدم هذا البرنامج فى إضافة بيانات حركة الصرف إلى ملف قاعدة البيانات الخاص بحركة الصرف للعملاء (Charges.dbf). ولكتابة هذا البرنامج يلزم أولا إنشاء شاشة تعديل بيانات الصرف (FNewChrg.fmt) وذلك بكتابة السطرين التاليين من مشيرة النقطة.

```
USE Charges
CREATE SCREEN FNewChrg
```

ويتم تحميل كل حقول ملف قاعدة البيانات (Charges.dbf) ما عدا حقلى الكمية (Amount) والدفع (Billed). وكما حدث فى شاشة إدخال العملاء (FNewCust) يتم حماية حقل رقم العميل من التعديل. وذلك باستخدام المفتاح (F10) وتحويل الإختيار (Action) إلى (Display/Say).

وعند الإنتهاء من تصميم الشاشة تصبح بالصورة الموضحة بالشكل (٢٠ - ٣). وهنا يجب ملاحظة أن إسم العميل وعنوانه لم يتم إضافته إلى الشاشة. حيث أنه لايكون ضمن حقول ملف حركة الصرف (Charges.dbf). ولكن فى العادة يحتاج المستخدم إلى معرفة إسم العميل وعنوانه أثناء تعديل البيانات الخاصة به. لذلك يتم إضافة متغيرات الذاكرة (M_Name) و (M_Address) إلى شاشة الإدخال. ولتنفيذ ذلك يتم استخدام الأمر (MODIFY COMMAND) فى تعديل برنامج تصميم الشاشة وذلك كالآتى :

```
MODIFY COMMAND FNewchrg.fmt
```

ثم يتم كتابة السطرين التاليين :

```
@ 4,33 SAY "Name:" + M_Name
@ 5,33 SAY "Address:" + M_Address
```

مع ملاحظة أن الإحداثيات يمكن تعديلها بناء على شكل الشاشة الذى سبق تصميمه.

Add New Payments

Customer Number : 999

Check Number : 9999999 Date : 99/99/99

Part Number : 9999 Qty: 99999.99 Unit Price: 99999.99

Description : XXXXXXXXXXXXXXXXXXXXXXXXXXXXX

Curser Movement by Up , Down , Left , and Right arrow keys

Insert Mode : Ins
Save : ^End or ^W

Delete Characters : Del
Abandon : ^Q

شكل (٢٠ - ٣)

وحتى يختبر مخطط البرامج الشاشة بعد إدخال هذا التعديل يمكنه كتابة الأوامر التالية من مشيرة النقطة (Dot Prompt).

```
M_name = "Mohamed"
M_Address = "12 Hegaz street"
USE Charges
SET FORMAT TO FNewChrg
APPEND
```

فإذا ظهرت الشاشة فإن هذا يدل على أن ملف التشكيل (Format File) سليم. وإذا لم تظهر فإن هذا يدل على وجود خطأ معين فى ملف التشكيل. ويلاحظ أنه تم إنشاء متغيرات الذاكرة (M_Name) و (M_Address) قبل إجراء الإختبار وذلك لأن هذه المتغيرات ليست موجودة ضمن حقول ملف حركة الصرف (Charges.dbf). أما أثناء تنفيذ برنامج الإضافة (NewChrg.prg) فإن هذه المتغيرات يتم إنشاؤها من خلال برنامج الخطوات (GetCust.prg) الذى سبق شرحه.

ويتم كتابة سطور برنامج الإضافة كالآتى :

```
* * * * * * * * * * * * * * NewChrg.prg.
* Add individual charges to the Charges database file.
* Called from AR main menu

* - - - - - Print the screen title
DO Title WITH "Enter New Charges"
* - - - - - Open both Charges and Customer databases.
SELECT 1
USE Customer INDEX CustNo , CustName
SELECT 2
USE Charges INDEX ChrgNo

* - - - - - Set up loop for adding entries.
Exiting = .F.
DO WHILE .NOT. Exiting
        * - - Get customer name or number , and validate.
```

```
        SELECT 1
        DO GetCust WITH M_Cust_No, M_Name, M_Address,Exiting
        * - - - Allow user to enter transaction data using FNewChrg screen.
        IF .NOT. Exiting
                SELECT 2
                APPEND BLANK
                REPLACE Cust_No WITH M_Cust_No
                REPLACE Date WITH DATE()
                REPLACE Billed WITH .F.
                SET FORMAT TO FNewChrg
                READ
                CLOSE FORMAT
                REPLACE Amount WITH Qty * Unit_Price
        ENDIF
        @ 3,0 CLEAR
ENDDO(While adding new transactions)
* - - - - - Return to main menu.
CLOSE DATABASES
RETURN
```

ملاحظة

إذا أريد إضافة الضريبة إلى الكمية (Amount) يمكن إضافة السطر التالى مثلا :

```
Amount = 1,05 * Amount
```

وذلك قبل إنتهاء الحلقة التكرارية.

٢٠ - ٤ برنامج إضافة حركة السداد (NewPay.prg)

يتم تشغيل هذا البرنامج عندما يختار المستخدم الرقم (٣) من القائمة الرئيسية للبرنامج (AR.prg). ويستخدم هذا البرنامج فى إضافة بيانات حركة السداد (Payments) إلى ملف قاعدة البيانات الخاص بحركة السداد (Payments.dbf).

وهذا البرنامج يماثل تماما برنامج إضافة حركة الصرف حيث يتم إنشاء شاشة إدخال بنفس الطريقة عن طريق كتابة السطرين التاليين :

```
USE Payments
CREATE SCREEN FNewPay
```

وفى هذه الحالة تظهر قوائم الإختيارات الخاصة برسم الشاشة. ويتم تحميل حقول ملف حركة السداد (Payments.dbf) كلها باستثناء حقل الترحيل (Posted). كما يتم استخدام مفتاح (F10) لتحويل حقل رقم العميل (Cust_No) إلى حالة (Display/Say) حتى يصبح محميا من أى تعديل بواسطة المستخدم.

ولإضافة المتغيرين (M_Name) و (M_Address) إلى شاشة الإدخال يتم استخدام نفس الطريقة السابقة. وذلك باستخدام الأمر (MODIFY COMMAND) عن طريق كتابة السطر التالى :

```
MODIFY COMMAND FNewPay.FMT
```

ثم يتم إضافة السطرين التاليين إلى برنامج رسم الشاشة.

```
@ 5,35 SAY "Name:" + M_Name
@ 6,35 SAY "Address:" + M_Address
```

وعند الإنتهاء من تصميم الشاشة تصبح بالصورة الموضحة بالشكل (٢٠ - ٤).

ويمكن استخدام هذه الشاشة فى إضافة أى حركة سداد (Payments). كما يمكن إضافة أى حركة ضبط (Adjustment) عند إرجاع العميل لأى صنف ويراد إضافته للرصيد. وفى هذه الحالة يتم ملء بيانات الشاشة مع تحديد الكمية التى تمت إعادتها. ثم يتم توضيح سبب رجوع هذه الكمية فى حقل الوصف (Descript).

Add New Payments

Customer Number : 999 **Name :**

Check Number : 9999999 **Address :**

Amount : 9999999.99

Description : XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

Curser Movement by Up , Down , Left , and Right arrow keys	
Insert Mode : Ins Save : ^End or ^W	Delete Characters : Del Abandon : ^Q

شكل (٢٠ - ٤)

وكما تم بالنسبة لبرنامج إضافة حركة الصرف يتم استخدام برنامج الخطوات (GetCust) فى البحث عن إسم العميل أو رقمه والتأكد من وجوده فى الملف. وكما سوف نلاحظ فإن البرنامج يماثل برنامج إضافة حركة الصرف (NewChrg.prg) مع اختلافات قليلة جدا. لذلك يمكن نسخه باستخدام الأمر (MODIFY COMMAND) وقراءة نفس هذا الملف فى الملف الجديد. وذلك بكتابة (^KR) ثم إجراء التعديلات المطلوبة فيه وذلك بدلا من كتابته من جديد. والسطور التالية توضح أوامر هذا البرنامج.

```
* * * * * * * * * * * * * NewPay.prg.
*    Add individual payments to the payments file.
*    Called from AR main menu

* - -Open both Payments and Customer databases.

SELECT 1
USE Customer INDEX CustNo , CustName
SELECT 2
USE Payments INDEX PayNo
```

```
* - - - - - Set up loop for adding entries.
Exiting = .F.
DO WHILE .NOT. Exiting
        * - - Get customer name or number , and validate.
        SELECT 1
        DO GetCust WITH M_Cust_No , M_Name , M_Address , Exiting
        * - Allow user to enter transaction data using FNewPay screen.
        IF .NOT. Exiting
                SELECT 2
                APPEND BLANK
                REPLACE Cust_No WITH M_Cust_No
                REPLACE Date WITH DATE()
                REPLACE Descript WITH "Payment"
                REPLACE Posted WIHT .F.
                SET FORMAT TO FNewPay
                READ
                CLOSE FORMAT
        ENDIF
        @ 3,0  CLEAR
ENDDO (While adding new transactions)
* - - - - - Return to main menu.
CLOSE DATABASES
RETURN
```

٢٠ - ٥ برنامج التعديل (AREdit.prg)

يتم تعديل حسابات العملاء باستخدام برنامج تعديل إضافة العملاء (EdCust.prg) وبرنامج تعديل حركة الصرف (EdChrg.prg) وبرنامج تعديل حركة السداد (EdPay.prg). ويتم التحكم فى هذه البرامج عن طريق برنامج قائمة التعديل (AREdit.prg). ويتم تشغيل هذا البرنامج عندما يختار المستخدم الرقم (5) من القائمة الرئيسية للبرنامج (AR.prg). وعند تشغيل هذا البرنامج تظهر القائمة الموضحة بالشكل (٢٠ - ٥).

```
                                    02/20/90   08:30:45
 Accounts Receivable Edit Menu

            1. Edit Customer File
            2. Edit Current Charges
            3. Edit Current Payments
            4. Return to main menu

  Enter choice ( 1 - 4 ) █
```

شكل (٢٠ - ٥)

كما يتم كتابة هذا البرنامج كالآتى :

```
* * * * * * * * * * * * * * * * * * * * * AREdit.prg
*   Menu for editing the A/R System.
*   Called from AR main menu
EChoice = 0
DO WHILE EChoice # 4
     CLEAR
     DO Title WITH "Accounts Receivable Edit Menu"
     TEXT
          1. Edit Customer File
          2. Edit Current Charges
          3. Edit Current Payments
          4. Return to main menu
     ENDTEXT
     @ 24,3 SAY "Enter choice (1 - 4) " ;
     GET EChoice PICT "9" RANGE 1,4
     READ
```

```
* - - - - -  Branch accordingly.
    DO CASE
        CASE EChoice = 1
            DO EdCust
        CASE EChoice = 2
            DO EdChrg
        CASE EChoice = 3
            DO EdPay
    ENDCASE
ENDDO (Echoice # 4)
* - - - - -  Return to main menu
RETURN
```

٢٠ - ٥ - ١ تعديل ملف العميل (EdCust.prg)

شاشة تعديل ملف العميل (FEdCust.scr) هى نفس شاشة إدخال العملاء (FNewCust.scr) مع اختلاف وحيد وهو أن العنوان يجب تغييره إلى (Edit Customers) ولذلك يتم نسخ ملف شاشة العميل فى الملف الجديد (FEdCust.scr) وذلك كالآتى :

```
COPY FILE FNewCust.scr TO FEdCust.scr
```

ثم يتم كتابة السطر التالى :

```
MODIFY SCREEN FEdCust
```

فى هذه الحالة تظهر السبورة (Blackboard) ويتم تعديل العنوان من (Add New Customers) إلى العنوان (Edit Customers) ثم تخزين الشاشة الجديدة. أنظر الشكل (٢٠ - ٦).

Edit Customers			
Customer Number : 999			
Customer Name :	XXXXXXXXXXXXXXXXXX		
Address :	XXXXXXXXXXXXXXXXXXXXXXX	Phone:	XXXXXXX
Starting Balance :	99999.99	Terms :	XXXXXXXXXXXXX
Balance(30 days):	99999.99	Balance(60 days):	99999.99
Balance(90 days):	99999.99	Balance(90 +days):	99999.99

شكل (٢٠ - ٦)

وبرنامج تعديل ملف العميل يسمى (EdCust.prg) وهو يستخدم برنامج الخطوات (GetCust.prg) فى البحث عن العميل برقمه أو باسمه مثل البرامج السابقة. ويتم كتابة سطور البرنامج كالآتى :

```
* * * * * * * * * * * * * * * * * * * * * * EdCust.prg
*   Edit Customers information
*   Called from A/R Edit menu
* - - - - -Print the screen title.

DO Title WITH "Enter Customers File"
USE Customer INDEX CustNo, CustName
* - - - - -  Set up loop for editing.
Exiting = .F.
SET DELETED OFF
DO WHILE .NOT. Exiting
        * - - - - -  GET Customer by name or number
        DO GetCust WITH M_Cust_No , M_Name , M_Address , Exiting
```

```
        * - Edit using the FEdCust screen (if not exiting)
        IF .NOT. Exiting
              SEEK M_Cust_No
              SET FORMAT TO FEdCust
              EDIT
              SET FORMAT TO
              @ 4,0  CLEAR
        ENDIF
ENDDO (While not exiting)
* - - - - -  Return to main menu
SET DELETED ON
CLOSE DATABASES
RETURN
```

٢٠ - ٥ - ٢   تعديل ملف الصرف (EdChrg.prg)

برنامج تعديل ملف الصرف (Charges.dbf) يسمح للمستخدم بإدخال رقم العميل أو إسمه لتحديد مكان هذا العميل فى الملف ثم استخدام مفتاحى (PgDn) و (PgUp) فى عرض كل حركة خاصة بهذا العميل للوصول إلى الحركة المطلوب تعديلها. وتستخدم الشاشة (FEdChrg.scr) فى تعديل حركة الصرف. ولإنشاء هذه الشاشة يتم نسخ شاشة إضافة الصرف (FNewchrg.scr) فى شاشة التعديل (FEdchrg.scr) وذلك بكتابة السطر التالى :

```
COPY FILE FNewChrg.scr TO FEdChrg.scr
```

ثم يتم كتابة السطر التالى :

```
MODIFY SCREEN FEdChrg
```

حتى يتم عرض السبورة (Blackboard) وتعديل عنوان الشاشة إلى (Edit Charges) كما يتم إضافة تعليمات أسفل الشاشة لتوضح للمستخدم كيفية الإنتقال من حركة إلى حركة أخرى باستخدام مفتاحى (PgUp) و (PgDn).

وحتى يظهر إسم العميل وعنوانه مع كل حركة يلزم فتح ملف العملاء (Customer.dbf) وربطه بملف الصرف وتثبيت مؤشر السجلات (Record Pointer) على السجل الخاص بهذا العميل ثم الحصول على إسم هذا العميل وعنوانه. ويتم ذلك عن طريق تعديل ملف الشاشة بواسطة الأمر :

```
MODIFY COMMAND FEdchrg.fmt
```

ثم يتم كتابة السطرين التاليين فى هذا الملف :

```
@ 4,33 SAY "Name:" + TRIM(A-> Cust_Name)
@ 5,33 SAY "Address:" + TRIM(A-> Address)
```

ويجب ملاحظة أن الإحداثيات الخاصة بهذين السطرين تتوقف على تصميم الشاشة. والشكل (٢٠ - ٧) يوضح صورة الشاشة المستخدمة فى تعديل حركة الصرف.

Edit Charges

Customer Number : 999 Name :
Address:
Invoice Number : 9999999 Date : 99/99/99
Part Number : 9999 Qty: 99999.99 Unit Price: 99999.99
Description : XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

Curser Movement by Up , Down , Left , and Right arrow keys	
Insert Mode : Ins Save : ^End or ^W	Delete Characters : Del Abandon : ^Q

شكل (٢٠ - ٧)

ولكى يتم التحكم فى مؤشر السجلات (Record Pointer) الموجود فى ملف العملاء للحصول على إسم العميل وعنوانه يجب إنشاء علاقة بين ملفى العملاء (Cust.dbf) والصرف (Charges.dbf) وذلك من خلال برنامج تعديل ملف الصرف (EdChrg.prg)

ويتضح ذلك من السطور التالية :

```
* - - - - Open both Charges and Customer databases.
SELECT 1
USE Customer INDEX CustNo, CustName
SELECT 2
USE Charges INDEX ChrgNo
* - - - - -  Set up relationship
SET RELATION TO Cust_No INTO Customer
```

وهذا الربط يجعل المؤشر يقف دائما على نفس السجل الخاص بالعميل بناء على رقم العميل بصرف النظر عن الحركة التى يتم عرضها على الشاشة. وبالتالى يظل إسم العميل وعنوانه معروضا على الشاشة أثناء تعديل أى حركة خاصة بهذا العميل.

والبرنامج يتم كتابة سطوره كالآتى :

```
* * * * * * * * * * * * * * * * * * * * * * EdChrg.prg
*    Edit invalid charges in the charges file.
*    Called from A/R Edit menu

* - -Print the screen title.
DO Title WITH "Edit Current Charges"
* - - - - -  Open both Customer and charges databases.
SELECT 1
USE Customer INDEX CustNo, CustName
SELECT 2
USE Charges INDEX ChrgNo
* - - - - -  Set up relationship
SET RELATION TO Cust_No INTO Customer
* - - - Set up memory variables and loop for editing
SET DELETED OFF
Exiting = .F.
```

```
DO WHILE .NOT. Exiting
        * - - - - -  Get Customer by name or number
        SELECT 1
        DO GetCust WITH M_Cust_No, M_Name, M_Address, Exiting
        * - Edit the transaction, if valid and not exiting
        IF .NOT. Exiting
                SELECT 2
                SEEK M_Cust_No
                IF FOUND .AND. .NOT. Billed
                        SET FORMAT TO FEdChrg
                        EDIT
                        CLOSE FORMAT
                ELSE
                        DO Error WITH "Already posted,Make adjustment"
                ENDIF
        ENDIF (not exiting)
        @ 4,0  CLEAR
ENDDO (While not exiting)

* - - - - -  Recalculate amount field.
SELECT 2
REPLACE ALL Amount WITH Qty * Unit_Price
SET DELETED ON
CLOSE DATABASES
* - - - - -  Return to main menu.
RETURN
```

ويلاحظ من البرنامج استخدام الجملة

```
IF FOUND() .AND. .NOT. Billed
```

حيث أنها تمنع تعديل السجل بعد ترحيله إلى الملف الرئيسي. أما إذا كان قد تم

ترحيله فتظهر رسالة خطأ (Error Message) توجه المستخدم إلى التعديل عن طريق إضافة حركة جديدة إلى ملف حركة الصرف. كما يلاحظ بعد انتهاء عملية التعديل إضافة السطر التالى :

```
REPLACE ALL Amound WITH Qty * Unit_Price
```

وهذا لضمان أن أى تعديل فى الكمية المصروفة أو فى سعرها يتم إدخاله مباشرة فى كمية الصرف الكلية لهذا العميل.

٢٠ - ٥ - ٣ تعديل ملف السداد (EdPay.prg)

هذا البرنامج يماثل تعديل ملف الصرف تماما مع بعض الإختلافات البسيطة. وهو يستخدم الشاشة (FEdPay.scr) فى التعديل ويتم إنشاؤها بنسخ الشاشة (FNewPay.scr) وذلك كالآتى :

```
COPY FILE FNewPay.scr TO FEdPay.scr
```

ثم يتم استخدام الأمر (MODIFY SCREEN) فى تحويل العنوان إلى (Edit Payments) وإضافة الإرشادات الخاصة باستخدام مفتاحى (PgUp) و (PgDn) فى الإنتقال من حركة إلى أخرى.

كما يتم استخدام الأمر (MODIFY COMMAND FEdpay.fmt) فى إدخال إسم العميل وعنوانه فى الشاشة. وذلك عن طريق كتابة السطرين التاليين فى ملف الشاشة (FEdpay.fmt).

```
@ 5,35 SAY "Name :" + TRIM(A -> Cust_Name)
@ 6,35 SAY "Address:" + TRIM(A -> Address)
```

والشكل (٢٠ - ٨) يوضح صورة شاشة الإدخال.

Edit Payments	
Customer Number : 999	**Name :**
Check Number : 9999999	**Address:**
Amoun : 99999.99	**Date : 99/99/99**
Description : XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX	
Curser Movement by Up , Down , Left , and Right arrow keys	
Insert Mode : Ins **Save : ^End or ^W**	**Delete Characters : Del** **Abandon : ^Q**

شكل (٢٠ - ٨)

وبرنامج التعديل يتكون من السطور التالية :

```
* * * * * * * * * * * * * * EdPay.prg
*   Edit invalid payments in the payments file.
*   Called from AR Edit menu
* - - - - -  Print the screen title.
DO Title WITH "Edit Current Payments"

* - - - Open both Customer and payments databases.
SELECT 1
USE Customer INDEX CustNo, CustName
SELECT 2
USE Payments INDEX PayNo

* - - - - -  Set up relationship
SET RELATION TO Cust_No INTO Customer

* - - - - -  Set up loop for editing entries
```

```
SET DELETED OFF
Exiting = .F.
DO WHILE .NOT. Exiting
        * - - - - -  Get Customer by name or number
        SELECT 1
        DO GetCust WITH ;
        M_Cust_No, M_Name, M_Address, Exiting
        * - - If not exiting , and transaction not already
        * - - - - -  posted , proceed with edit.
        IF .NOT. Exiting
                SELECT 2
                SEEK M_Cust_No
                IF FOUND() .AND. .NOT. Posted
                        SET FORMAT TO FEdPay
                        EDIT
                        CLOSE FORMAT
                ELSE
                        DO Error WITH  "Already posted, Make adjustment:"
                ENDIF
        ENDIF (not exiting)
        @ 4,0  CLEAR
ENDDO(while not exiting)
* - - - - -  Close databases and return to edit menu
SET DELETED ON
CLOSE DATABASES
RETURN
```

الفصل الحادي والعشرون

تقارير برنامج حسابات العملاء

يعتمد برنامج حسابات العملاء (A/R) على مجموعة من التقارير (Reports) مثل الفواتير (Invoices) والتقارير الشهرية والتقارير المختصرة (Summary reports) والتقارير الزمنية (Aging reports) التى توفر للمستخدم المتابعة الدقيقة للبيانات وتقديم الخدمة السريعة للعميل. وكذلك التقارير التاريخية (Historical Reports) التى تساعده على اتخاذ القرارات وإجراء التعديلات المطلوبة فى النظام بناء على دراسات إحصائية للمخرجات.

وفى الواقع فإن كتابة البرامج التى توفر هذه الأنواع من التقارير تحتاج إلى كثير من الوسائل المتقدمة فى كتابة البرامج وخصوصا عندما يراد الإحتفاظ بسرعة تشغيل البرنامج وكفاءته. ولذلك فسوف يتم استخدام برامج الخطوات (Procedures) والمعاملات (Parameters) فى بعض البرامج كما سيتم تصميم برامج لأنواع مختلفة من التقارير.

٢١ - ١ برنامج قائمة التقارير الرئيسية (ARPrint.prg)

عندما يختار المستخدم الرقم (4) من القائمة الرئيسية لبرنامج حسابات العملاء (A/R) تظهر شاشة الإختيارات الموضحة بالشكل (٢١ - ١).

```
A/R Print Menu

1. Print monthly statements
2. Print monthly summay
3. Print aging reports
4. Do quick looking of customer status
5. Review history
6. Retrun to main menu
Enter choice ( 1 - 6 ) ■
```

شكل (٢١ - ١)

والإختيار الأول يستخدم عندما يراد طباعة الفواتير (Invoices). ويجب تشغيله مرة واحدة فى الشهر عندما يراد ترحيل الحسابات إلى الملف الرئيسى.

والاختياران (2 , 3) يستخدمان أيضا مرة واحدة فى الشهر بعد طباعة الفواتير للحصول على تقرير مختصر عن موقف الفواتير الخاصة بكل عميل.

والاختيار رقم (4) يستخدم عندما يريد العميل مراجعة الحساب الخاص به.

والاختيار رقم (5) يوفر مراجعة أى بيانات تاريخية لأى حساب لمساعدة المستخدم على متابعة المسير التاريخى لأى عميل وتوفير المعلومات التى تتعلق بفترة معينة محصورة بين تاريخين وكذلك توفير معلومات عن حالة بيع صنف معين.

والبرنامج الخاص بعرض القائمة الرئيسية للتقارير (ARPrint.prg) لايختلف عن أى برنامج من برامج القائمة الرئيسية الأخرى. ويتكون من السطور التالية :

```
* * * * * * * * * * * * * * * * * * * * * * ARPrint.prg
*   Menu of Print options for the A/R system.
*   Called from A/R main menu

PChoice = 0
SET DELETED ON
DO WHILE PChoice # 6
        CLEAR
        DO Title WITH "A/R Print Menu"
        TEXT
                1. Print monthly statements
                2. Print monthly Summary
                3. Print aging reports
                4. Do quick looking of customer status
                5. Review history
                6. Return to main menu
        ENDTEXT
        @ 24,1 SAY "Enter choice (1 - 6)" ;
        GET Pchoice PICT "9" RANGE 1,6
        READ
```

```
*----- Branch accordingly.
DO CASE
        CASE PChoice = 1
                DO Bills
        CASE PChoice = 2
                RepForm = "ARSumm"
                DO AgeSumm
        CASE PChoice = 3
                RepForm = "Aging"
                DO AgeSumm
        CASE PChoice = 4
                DO ARStat
        CASE PChoice = 5
                DO ARHist
ENDCASE
ENDDO(PChoce # 6)
*----- Return to main menu
SET DELETED OFF
RETURN
```

ويلاحظ من هذا البرنامج أن الاختيار (2) والاختيار (3) يؤديان إلى تنفيذ نفس البرنامج (AgeSumm) ولكن البرنامج يعطى تقريرا مختلفا فى كل حالة كما سيتم الإيضاح فيما بعد.

٢١ - ٢ ملف الخطوات (BillProc.prg)

يلاحظ من البرنامج السابق الخاص بقائمة التقارير وجود برنامج الفواتير الشهرية (Bills.prg) الخاص بالاختيار (1) وبرنامج تحديد الحالة (ARStat.prg) الخاص بالاختيار (4) وهما يستخدمان فى طباعة الفواتير (Invoices). ولزيادة سرعة تشغيل البرنامج وكفاءته يقوم كل من هذين البرنامجين بتشغيل ملف خطوات. وهذا الملف يتم تسميته (BillProc.prg). وهو يحتوى على برنامج الخطوات (PrintBills.prg) بالإضافة إلى برنامج آخر يسمى (RowCheck) وهو يقوم باختبار رقم السطر فإذا زاد

عدد السطور عن طول الصفحة يتم الإنتقال إلى الصفحة التالية. وقد كان يمكن كتابة هذه البرامج كلها فى ملف الخطوات (Proclib1) السابق إنشاؤه ولكن ذلك سوف يتطلب استخدام معالج كلمات آخر غير معالج الكلمات المستخدم مع (DBase III+). وليست هناك حاجة لهذا فى برنامج حسابات العملاء. حيث يمكن فتح ملف الخطوات المطلوب وقت الحاجة ثم إغلاقه بعد ذلك. ولذلك فإن تعدد ملفات الخطوات لايؤثر فى كفاءة البرنامج.

والملف (BillProc.prg) يتكون من السطور التالية :

```
* * * * * * * * * * * * * * * * * * * * * * * BillProc.prg
* Print a bill,using procedures PrintBills and
* Rowcheck Called from Bills.prg and ARstat.prg
PROCEDURE PrintBills
PARAMETERS M_Cust_No , Printer, Status

* - - - IF printer , print address, date in English
CLEAR
Page = 1
IF Printer
       EngDate = CMONTH (DATE()) + STR(DAY(DATE()), 3) + ;
            "," + STR(YEAR(DATE()), 4)
       @ 1,0  SAY "My company , inc."
       @ 1,60 SAY EngDate
       ROW = 6
ELSE
       ROW = 1
ENDIF

* - - - -Print Customer name and address Using
* - - - -Row variable to control display and eject.
SELECT 1
IF Cus_Name # "  "
       @ Row,0 SAY Cus_Name
```

```
ENDIF
@ Row+1,0 SAY Address
Row = Row + 4

* - - -Print customer number , terms and statrting
* - - -balance from the customer file .
@ Row,0 SAY "Customer No, : " + STR(Cust_No, 4)
@ Row+1,0 SAY "Terms  :" + Terms
@ Row+2,0 SAY "Balance of " + DTOC(LAST_UPDAT) + ":"
@ Row+2,25 SAY Start_Bal PICT "999,999.99"
@ Row+3,0 SAY Uline
Start = Start_Bal
Row = Row + 5

* - - - - -  Print heading for charges .
@ Row,0 SAY "Inv. #  Part  Description    QTY"
@ Row,43 SAY "Price   Total   Date "

Row = Row + 2
* - - Select charges database :list and total current
* - - charges.
SELECT 2
SEEK M_Cust_No
Tot_Charge = 0
DO WHILE Cust_No = M_Cust_No .AND. .NOT. EOF()
        IF Status .OR. .NOT. Billed
                @ ROW,0 SAY Invoice_No
                @ ROW,7 SAY Part_No
                @ ROW,13 SAY Descript
                @ ROW,34 SAY Qty
                @ ROW,39 SAY Unit_Pric PICT "999,999.99"
                @ ROW,50 SAY Amount PICT "999,999.99"
```

```
            @ ROW,62 SAY Date
            Tot_Charge = Tot_Charge + Amount
            ROW = ROW + 1
            * - - - - Make as billed if not a status check
            IF .NOT. STATUS
                        REPLACE Billed WITH .T.
            ENDIF
      ENDIF(status report and not already billed)
      SKIP
ENDDO (Cust_No = M_Cust_No)
* - - - - -  Print payments heading .
@ ROW+1,0 SAY Uline
@ ROW+2,1 SAY "Payments / Adjustments"
ROW = ROW + 4

* - - - Check row position if displayed on screen.
DO RowCheck WITH 20,64
* - - - Select Payments file : list and total
* - - - payments/ adjustments.
SELECT 3
SEEK M_Cust_No
Tot_Pay = 0
DO WHILE Cust_No = M_Cust_No .AND. .NOT. EOF()
      IF Status .OR. .NOT. Posted
            @ Row,0 SAY "Check #"
            @ Row,9 SAY Check_No
            @ Row,16 SAY Descript
            @ Row,50 SAY Amount PICT "999,999.99"
            @ Row,62 SAY Date
            Tot_Pay = Tot_Pay + Amount
            ROW = ROW + 1
```

```
            * - - Check row position if displayed on screen
            DO RowCheck WITH 20,64
            * - - - Mark as billed if not status report
            IF .NOT. Status
                        REPLACE Posted WITH .T.
            ENDIF
      ENDIF(status report and not already posted)
      SKIP
ENDDO(while Cust_No = M_Cust_No)

* - - - Check row position if displayed on screen.
@ Row,0 SAY Uline
DO RowCheck WITH 17,56

* - - Print starting balance ,total charges, payments,
* - - ending balance , and thank you note.
SET FIXED ON
@ Row+1,5 SAY "Previous balance :"
@ Row+1,25 SAY START PICT "999,999.99"
@ Row+2,5  SAY "Total charges :"
@ Row+2,25 SAY Tot_Charge PICT "999,999.99"
@ Row+3,5  SAY "Payments received"
@ Row+3,25 SAY Tot_Pay PICT "999,999.99"
@ Row+4,5  SAY "Balance due   :"
@ Row+4,25 SAY (Start + Tot_Charge) - Tot_Pay  PICT "999,999.99"
SET FIXED OFF
IF Printer .AND. Status
      @ Row+8,10 SAY * * *  Duplicate Invoice * * *
ENDIF
IF Pritner .AND. .NOT. Status
      @ Row+8,5  SAY "Thank you"
ENDIF
```

```
*----- Pause if not going to the printer.
IF .NOT. Printer
      @ 22,0 CLEAR
      WAIT
ENDIF

*----- Done printing bill, Return to menu.
RETURN

*--- Procedure for checking row positions on screen or printer
PROCEDURE RowCheck
PARAMETERS ScreenMax, PrintMax
IF .NOT. Printer .AND. Row >= ScreenMax
      @ 23,0 CLEAR
      WAIT "Press any key for next page"
      Row = 1
      CLEAR
ENDIF(Row too big for screen)
IF Printer .AND. Row >= PrintMax
      @ Row+2,70 SAY "Page" + STR(Page, 1)
      Page = Page + 1
      EJECT
      ROW = 5
ENDIF (Row too big for printer)
RETURN
```

ويلاحظ فى البرنامج استخدام متغير الذاكرة (Status) لطباعة تقارير الحالة (Status) دون أن يؤثر ذلك على الفاتورة الشهرية. فإذا تم إدخال القيمة (.T.) إلى المتغير (Status) فإن هذا يعنى أن المطلوب تقرير حالة فقط وليس تقريرا شهريا ولذلك تتم طباعة التقرير دون ترحيل. أما إذا كان التقرير فاتورة شهرية (Not Status) فيتم طباعته مع تحويل حقل الترحيل إلى (False) حتى يتم ترحيله بعد ذلك باستخدام الاختيار (6) من القائمة الرئيسية لبرنامج حسابات العملاء.

ويجدر العلم أن الملف السابق ما هو إلا ملف خطوات يتم استدعاؤه بواسطة برنامج التقارير الشهرية (Bills.prg) وبرنامج تحديد الحالة (ARStat.prg). وهذان البرنامجان سيتم شرحهما فى الأجزاء التالية.

٢١ - ٣ برنامج الفواتير الشهرية (Bills.prg)

يستخدم هذا البرنامج عند إدخال الإختيار رقم (1) فى قائمة برنامج طباعة تقارير حسابات العملاء. وهذا البرنامج يقوم بطباعة الفواتير الشهرية وفى نفس الوقت يقوم بإدخال القيمة (.F.) فى حقل الترحيل سواء كان حقل التسديد (Billed) فى ملف الصرف (Charges.dbf) أو حقل الترحيل (Posted) فى ملف السداد (Payments.dbf). وذلك حتى يتسنى للمستخدم بعد ذلك ترحيل هذه الفواتير إلى الملف الرئيسى عن طريق برنامج الترحيل الشهرى (Monthly Posting) بالإختيار رقم (٦) من القائمة الرئيسية لبرنامج حسابات العملاء.

ويتكون هذا البرنامج من السطور التالية :

```
************************ *Bills.prg
*   Prints monthly statemetns.
*   Called from AR Print menu.

* - - - - - Have user prepare printer (or cancel).
Proceed = " "
DO Title WITH "Print Monthly Bills "
@ 15,5 SAY "Repare printer and press a key to proceed"
@ 17,5 SAY "(Type X to cancel)" GET Proceed PICT "!"
READ
* - - - - - Return to menu if requested.
IF Proceed = "X"
   RETURN
ENDIF

* - - - Open files and delete records with "0" Amount.
```

```
SELECT 1
USE Customer INDEX CustNo
SELECT 2
USE Charges INDEX ChargNo
DELETE ALL FOR Amount = 0
SELECT 3
USE Payments INDEX PayNo
DELETE ALL FOR Amount = 0 ,

* - - - Set decimal place to 2 , send @ ... SAY to
* - - - printer and open BillProc procedure file.
CLEAR
SET DECIMALS TO 2
SET DEVICE TO PRINT
SET PROCEDURE TO BillProc

* - Set parameters to printer and "not status" report.
Printer = .T.
Status = .F.

* - - - Loop through Customer database and print a
* - - - bill for every one
SELECT 1
DO WHILE .NOT. EOF()
        Lookup = Cust_No
        DO PrintBills WITH Lookup , printer , status
        EJECT

        * - - - - -  Set next customer
        SELECT 1
        SKIP
ENDDO
```

```
* - - - - - Done , Close files
SET DEVICE TO SCREEN
CLOSE DATABASES
CLOSE PROCEDURE
* - - - - - Open ProcLib1 procedure file
SET PROCEDURE TO ProcLib1
* - - - Print reminder about posting , then return to
* - - - main menu .
CLEAR
TEXT
        monthly postings (main menu option 5) should be
        performed immediately after printing the monthly
        statements.
ENDTEXT
?

@ 22,10 SAY "Press any key to return to main menu..."
WAIT " "
RETURN TO Master
```

ويلاحظ فى هذا البرنامج استخدام ملف الخطوات (BillProc.prg) لطباعة التقارير الشهرية. كما يلاحظ فتح ملف الخطوات الآخر (ProcLib1) قبل نهاية البرنامج حتى يستخدم فى باقى البرامج التى سوف تحتاجه. كما يلاحظ استخدام الأمر (RETURN TO MASTER) للرجوع إلى القائمة الرئيسية لبرنامج حسابات العملاء مباشرة حتى يتسنى للمستخدم استخدام الإختيار رقم (6) فى ترحيل الفواتير.

٢١ - ٤ برنامج اختبار الحالة (ARStat.prg)

يستخدم هذا البرنامج عند إدخال الإختيار رقم (٤) فى قائمة برنامج طباعة تقارير حسابات العملاء. وهو يسمح للمستخدم باختبار حالة العميل من حيث تسديد الفواتير (Billed) أو عدم تسديدها. وهو يتيح له الحصول على صورة أخرى من الفاتورة المرسلة إلى العميل لتسديدها. ويلاحظ فى هذا البرنامج إعطاء المتغير (Status) القيمة (True)

حتى يعرف برنامج الخطوات (PrintBills) أن المطلوب هو اختبار الحالة فقط وليس الفاتورة الحقيقية. وهذا البرنامج يسمى (ARStat.prg) ويتكون من السطور التالية :

```
********************** ARStat.prg
*   Quick lookup of a single statement.
*   Called from A/R Print menu.

*-- Open files and delete records with "0" amounts.
SELECT 1
USE Customer INDEX CustNo
SELECT 2
USE Charges INDEX ChrgNo
DELETE ALL FOR Amount = 0
SELECT 3
USE Payments INDEX PayNo
DELETE ALL FOR Amount = 0

*--- Set up memory variables for status report.
Status = .T.
Printer = .F.
M_Cust_No = 0
M_Name = "  "
Exiting = .F.
DO WHILE .NOT. Exiting
        *----- Print screen title
        DO Title WITH "Quick lookup of current status"
        *----- Get Customer by number or name
        SELECT 1
        DO GetCust WITH ;
        M_Cust_No , M_Name, M_Address, Exiting

        *----- Proceed with bill.
        IF .NOT. Exiting
```

```
            *- - - - -  Ask about printer.
            @ 5,0  CLEAR
            LP = " "
            @ 15,5 SAY "Send statement to printer ? Y/N ;
                      GET LP PICT "!"
            READ
            CLEAR
            *- - - - -  Set up printer if necessary
            IF LP = "Y"
                  Printer = .T.
                  SET DEVICE TO PRINT
            ENDIF

            * - - -Print current statement for customer.
            SET PROCEDURE TO BillProc
            DO PrintBill WITH M_Cust_No , Printer, STATUS
            CLOSE PROCEDURE
            SET PROCEDURE TO ProcLib1

            * - - - - -  turn off printer
            IF Printer
                  EJECT
                  SET DEVICE TO SCREEN
            ENDIF
      ENDIF (not exiting)
ENDDO(while not exiting)
* - - - Close files and return to main menu
CLOSE DATABASES
RETURN
```

٢١ - ٥ برنامج التقارير المختصرة والزمنية (AgeSumm.prg)

الإختياران (2) و (3) من قائمة برنامج طباعة تقارير حسابات العملاء يسمحان للمستخدم بالحصول على تقارير شهرية مختصرة وتقارير زمنية (Aging Reports). وهذه التقارير يتم إنشاؤها عادة بعد طباعة الفواتير الشهرية مباشرة.

والتقارير الشهرية المختصرة (Summary Reports) تعرض الموازنة الحالية للعميل وموقف الصرف والسداد الخاص به. حيث يتم عرض تقرير بالصورة الموضحة بالشكل (٢١ - ٢).

ويتم إنشاء صورة التقرير باستخدام الأمر (CREATE REPORT) وذلك كالآتى :

```
USE Customer
CREATE REPORT ARSumm
```

Page No 1 — **Monthly Activity Summary** — 11/21/90

Cust. No.	Name	Current Balance	Current Charges	Current Payments	Last Update
1000	Ahmed Salem	80.00	180.00	100.00	02/30/90
1021	Tarek Fathy	0.00	0.00	150.00	01/10/90
** Total **		80.00	180.00	250.00	

شكل (٢١ - ٢)

وفى هذه الحالة تظهر قوائم برنامج المساعد (Assistant) التى يتم من خلالها تحديد محتويات الأعمدة (Columns). وذلك كما هو موضح بالشكل (٢١ - ٣).

أما التقارير الزمنية فتسمح للمستخدم بعرض الموازنة الخاصة بكل عميل فى فترات مختلفة لمراقبة موقف هذا العميل حيث يكون التقرير بالصورة الموضحة بالشكل

(٢١ - ٤).

ولإنشاء هذا التقرير يتم استخدام الأمر (CREATE REPORT) كالآتى :

```
USE Customer
CREATE REPORT Aging
```

وفى هذه الحالة تظهر قوائم برنامج المساعد (Assistant) التى يتم من خلالها تحديد محتويات الأعمدة (Columns) وذلك كما هو موضح بالشكل (٢١ - ٥)

Column	Contents	Heading	Width	Decimal	Total
1	Cust_No	Cust No	5	0	N
2	Trun[Cust_Name]	Name	26	0	
3	Start_Bal	Current : Balance	7	2	Y
4	Chg_Curr	Current : Change	8	2	Y
5	Pay_Curr	Current : Payments	8	2	Y
6	Last_Update	Last : Posted	8	0	

شكل (٢١ - ٣)

Page No 1 02/30/90

Accounts Receivable Aging Report

Name	Current Balance	30 days	60 days	90 days	90 +days
Customer Number 1000 Ahmed Salem	80.00	88.00	35.00	0.00	0.00
Customer Number 1020 Tarek Fathy	0.00	98.00	90.00	130.00	200.00

شكل (٢١ - ٤)

Column	Contents	Heading	Width	Decimal	Total
1	Trun[Cust_Name]	Name	20		
2	Start_Bal	Current : Balance	9	2	N
3	Bal_30	30 days	8	2	N
4	Bal_60	60 days	8	2	N
5	Bal_90	90 days	8	2	N
6	Bal_90Plus	90 + days	8	2	N

شكل (٢١ - ٥)

ولعرض رقم العميل مع إسمه يستخدم الاختيار (Group) ويتم استخدام رقم العميل حقلا للتجميع بناء عليه وكتابة عنوان المجموعة (Customer Number) وبذلك يظهر هذا العنوان يليه رقم العميل مع كل عميل كما يتضح من صورة التقرير السابق إيضاحها.

ويتم طباعة التقارير المختصرة (Summary Reports) والتقارير الزمنية (Aging Reports) باستخدام البرنامج (AgeSum.prg) مع استخدام الماكرو لتحديد نوع التقرير المطلوب. ويتضح ذلك من الأوامر التالية الموجودة فى برنامج الطباعة (ARPrint.prg).

```
CASE PChoice = 2
      RepForm = "ARSumm"
      DO AgeSumm
CASE PChoice = 3
      RepForm = "Aging"
      DO AgeSumm
```

وبعد اختيار المستخدم للرقم (2) يتم تخزين كلمة (ARSumm) فى متغير الذاكرة (RepForm). وعند اختياره للرقم (3) يتم تخزين كلمة (Aging) فى نفس هذا المتغير. وعند تنفيذ البرنامج (AgeSum) فى الحالتين يتم طباعة التقرير الخاص بكل حالة.

ويتم كتابة سطور هذا البرنامج كالآتى :

```
* * * * * * * * * * * * * * * * * * * * *AgeSumm.prg
*   Print aged balances or summary reports.
*   Called from A/R Print menu.
@ 5,0 CLEAR

* - - - - - Ask about printer
STORE "  " TO LP, Mac
@ 15,5 SAY "Send report to printer ? (Y/N)"  GET LP PICT "!"
READ
* - - - - - Set up the printer if necessary.
IF LP = "Y"
      Mac = "TO PRINT"
      WAIT "Prepare printer,then press any key to print"
ENDIF

* - - - - Use Customer database and report stored in
* - - - - RepForm
CLEAR
USE Customer INDEX CustNo
REPORT FORM & RepForm & Mac

* - - - - Pause , if necessary , then return to menu
IF LP # "Y"
      WAIT "Press any key to return to AR Print menu."
ELSE
      EJECT
INDIF
RETURN
```

وعند تنفيذ هذا البرنامج يتم التعويض بنوع التقرير المطلوب سواء كان تقريرا مختصرا

(ARSumm) أو زمنيا (AgeSumm) مكان المتغير (RepForm) حسب اختيار المستخدم. كما يتم التعويض بالسلسلة الحرفية (TO PRINT) مكان المتغير (Mac) إذا اختار المستخدم الطباعة. وهذا يؤدى إلى أن يصبح الأمر فى الصورة التالية :

```
REPORT FORM ARSumm TO PRINT
```

وذلك فى حالة اختيار المستخدم للاختيار رقم (2) من قائمة الطباعة. وبالتالى يتم طباعة التقرير المطلوب.

٢١ - ٦ التقارير التاريخية (ARHist.prg)

التقارير التاريخية تسمح للمستخدم بالبحث عن أى بيانات سابقة للعميل خلال أى فترة زمنية. ويتم الدخول فى قائمة التقارير التاريخية عندما يختار المستخدم الاختيار رقم (5) من القائمة الرئيسية لبرنامج طباعة تقارير حسابات العملاء. وفى هذه الحالة تظهر القائمة الموضحة بالشكل (٢١ - ٦).

```
History Menu                    02/20/90    08:30:45

        1. Search by Customer Code
        2. Search by Product Code
        3. Search by Data
        4. Return to main menu

  Enter choice ( 1 - 4 )
```

شكل (٢١ - ٦)

وعندما يختار المستخدم الاختيار رقم (1) من القائمة ثم يقوم بكتابة رقم العميل المطلوب تظهر جميع البيانات التاريخية الخاصة بهذا العميل بالصورة الموضحة بالشكل (٢١ - ٧). ويلاحظ من هذا الشكل أن التقرير مكون من جزئين جزء علوى وجزء سفلى. وهذا يتطلب إنشاء صورتين للتقرير باستخدام الأمر (CREATE REPORT). حيث يتم طباعة الجزء العلوى من التقرير باستخدام الملف (CusHist1.frm) ويتم إنشاؤه عن طريق كتابة السطرين التاليين :

```
USE ChrgHist
CREATE REPORT CusHist1
```

ويلاحظ هنا استخدام الملف (ChrgHist.dbf) الذى سبق إنشاؤه قبل كتابة برنامج حسابات العملاء.

وفى هذه الحالة تظهر قوائم برنامج المساعد (Assistant) التى يتم عن طريقها تحديد عنوان التقرير (Customer History). كما يتم تحديد محتويات الأعمدة كالآتى :

(Check_No) ، (Part_No) ، (Qty) ، (Descript) ، (Unit_Price) ، (Amount) ، (Date) .

كما يتم تحديد عناوين الأعمدة كما يتضح من الشكل (٢١ - ٧).

Page No 1 — **Customer History** — 11/21/90

Cust No.	Part No.	Qty	Description	Uint Price	Total	Date
1000	BBB	5	Floppy Disks	1600	80.00	01/30/90
1000	AAA	2	Printer	900	1800.00	02/05/90

Cust No.	Check No.	Description	Amount	Date
1000	1333	Payment	100.00	02/06/90
1000	1750	Payment	200.00	02/29/90

شكل (٢١ - ٧)

أما الجزء السفلى من التقرير فهو يتعلق بموقف التسديد الخاص بهذا العميل. ويتم تكوينه بواسطة الملف (CusHist2.frm) الذى يتم إنشاؤه بكتابة السطرين التاليين.

```
USE PayHist
CREATE REPORT CusHist2
```

وفى هذه الحالة تظهر قوائم برنامج المساعد كما سبق الإيضاح ويتم من خلالها تحديد محتويات الأعمدة كالآتى :

(Cust_No) ، (Check_No) ، (Descript) ، (Amount) ، (Date)

كما يتم تحديد عناوين هذة الأعمدة كما يتضح من الشكل (٢١ - ٧).

وإذا اختار المستخدم الاختيار رقم (2) من قائمة التقارير التاريخية فإن هذا يعنى أن التقرير المطلوب عن صنف معين. ولذلك يتم إدخال رقم هذا الصنف (Part_no). وفى هذه الحالة يظهر التقرير الموضح بالشكل (٢١ - ٨).

Page No 1 | **Monthly Activity Summary** | 11/21/90

Part No.	Description	Qty	Uint Price	Total	Date
AAA	Printer	2	900	1800.00	02/05/90
AAA	Printer	3	900	2700.00	02/10/90
AAA	Printer	5	900	4500.00	02/20/90
Total		**10**	**2700**	**8200.00**	

شكل (٢١ - ٨)

ويتم إنشاء هذا التقرير عن طريق كتابة السطرين التاليين :

```
USE BillHist
CREATE REPORT CodeHist
```

وعند ظهور قوائم الاختيارات الخاصة ببرنامج المساعد (Assistant) يتم كتابة عنوان التقرير (Heading) كالآتى :

(Product Code History)

كما يتم تحديد محتويات الأعمدة (Columns) كالآتى :

(Part_No) ، (Descript) ، (Qty) ، (Unit_Price) ، (Amount) ، (Date)

ويفيد هذا التقرير فى متابعة موقف كل صنف ومعرفة معدل صرفه.

وإذا اختار المستخدم الرقم (3) من قائمة التقارير التاريخية فإن هذا يعنى أن المستخدم يريد طباعة تقرير عن موقف الموازنة فى تاريخ محدد. وفى هذه الحالة يقوم بإدخال التاريخ المطلوب.

وهذا التقرير يتكون من جزئين. الجزء العلوى يتم إنشاؤه بواسطة الملف (DatHist1.frm) ويتم ذلك عن طريق كتابة السطرين التاليين :

```
USE BillHist
CREATE REPORT DatHist
```

ويتم كتابة العنوان (Product Code History) للجزء العلوى كما يتم تحديد محتويات الأعمدة (Columns) كما يلى :

(Date) ، (Part_No) ، (Descript) ، (Qty) ، (Unit_Price) ، (Amount) ، (Cust_No)

والجزء السفلى يتم إنشاؤه بواسطة الملف (DatHist2.frm). ويتم ذلك بكتابة السطرين التاليين :

```
USE PayHist
CREATE REPORT DatHist2
```

ويتم تحديد محتويات الأعمدة كالآتى :

(Date) ، (Check_No) ، (Amount) ، (Descript) ، (Cust_No) .

ويجب ملاحظة أن هذه المجموعة من التقارير هى مجرد إقتراحات. ويستطيع مخطط البرامج إنشاء أى تقارير أخرى حسب الحاجة. وبعد إنشاء التقارير المختلفة يتم كتابة البرنامج الذى يؤدى إلى طباعة كل نوع من هذه التقارير. وذلك كالآتى :

```
* * * * * * * * * * * * * * * * * * * * * * ARHist.prg,
*  Search history, and current charges and payments
*  files and display summary data.
*   Called from AR Print Options menu.
HChoice = 0
DO WHILE HChoice # 4
        CLEAR
        DO Title WITH "History Menu"
        TEXT
                1. Search by customer code
                2. Search by product code
                3. Search by date
                4. Return to main menu.
        ENDTEXT
        @ 24,1 SAY "Enter choice (1 - 4) "
          GET HChoice PICT "9" RANGE 1,4
        READ
        * - - - - - - Set up search macro accordingly.
        @ 4,0  CLEAR
        DO CASE
                CASE HChoice = 1
                        M_Cust_No = 0
                        @ 15,5 SAY "Enter customer number" ;
                               GET M_Cust_No PICT "99999"
```

```
                READ
                LookAT = "Cust_No"
                LookFOR = M_Cust_No
                RepForm1 = "CusHist1"
                RepForm2 = "CusHist2"
                SET EXACT ON
        CASE HChoice = 2
                M_Code = SPACE (5)
                @ 15,5 SAY "Enter product code" ;
                    GET M_Code
                LookAT = "UPPER(Part_No)"
                LookFOR = UPPER(M_Code)
                RepForm1 = "CodeHist"
                SET EXACT ON
        CASE HChoice = 3
                M_Dat = SPACE (8)
                @ 15,5 SAY "Enter Date " GET M_Date
                READ
                LookAT = "DTOC(Date)"
                LookFOR = TRIM(M_Date)
                RepFlrm1 = "DatHist1 "
                RepForm2 = "DatHist2"
                SET EXACT OFF
        CASE Hchoice = 4
                SET EXACT OFF
                RETURN
ENDCASE
* - - - - - Ask about printer.
@ 5,0 CLEAR
STORE " " TO Lp, Mac
@ 15,5 SAY "Send report to printer ? (Y/N)" ;
   GET LP PICT "!"
```

```
READ
IF Lp = "Y"
        Mac = "TO PRINT"
ENDIF
* - - - - - Search billing History file
USE BillHist
SET FILTER TO &LookAt = LookFor
COPY TO Temp
USE Temp
APPEND FROM Charges FOR & LookAT = LookFOR
SET FILTER TO &LookAT = LookFOR
REPORT FORM &RepForm1 &Mac

* - - - if not searching for product code , Search
* - - - Payments files
IF HChoice # 2
        USE PayHist
        SET FILTER TO &LookAT = LookFOR
        REPORT FORM &RepForm2 &Mac PLAIN NOEJECT
ENDIF

* - - - IF report not going to printer, pause.
IF Lp # "Y
        ?
        ?
        WAIT
ENDIF
ENDDO(HChoice # 4)
```

الفصل الثانى والعشرون

التحديث الشهرى للنظام

يتم تحديث النظام مرة واحدة كل شهر عندما يختار المستخدم الرقم (6) من القائمة الرئيسية لبرنامج حسابات العملاء حيث يتم تشغيل برنامج الترحيل الشهرى للنظام (Post.prg) الذى يقوم بالآتى :

١ - نقل بيانات الموازنة خلال ثلاثين يوما أو ستين يوما أو ٩٠ يوما أو أكثر من ذلك فى بيان الفترة السابقة على الترتيب وذلك باستخدام الأمر (REPLACE) بحيث تنتقل بيانات حقل الموازنة خلال (٣٠) يوما إلى حقل الموازنة خلال (٦٠) يوما وبيان حقل الموازنة خلال (٦٠) يوما إلى حقل الموازنة خلال (٩٠) يوما وتضاف بيانات حقل الموازنة خلال (٩٠) يوما إلى بيانات حقل الموازنة خلال أكثر من ٩٠ يوما.

٢ - جميع فواتير حركة الصرف التى تمت كتابتها يتم تلخيصها وإدخالها فى ملف قاعدة بيانات جديد يسمى (Summary.dbf). ثم يتم تحديث الملف الرئيسى (Master.dbf) من الملف (Summary.dbf) باستخدام الأمر (UPDATE FROM).

٣ - كل السجلات الخاصة بالفواتير التى تم تسديدها (Billed = .T.) تضاف إلى ملف الصرف التاريخى (BillHist.dbf) وتمسح من ملف حركة الصرف (Charges.dbf) وذلك لتجهيز ملف حركة الصرف للشهر الجديد.

٤ - يتم تنفيذ نفس هذه العملية على ملف حركة التسديد (Payments.dbf). حيث يتم نقل جميع السجلات التى تم ترحيلها (Posted = .T.) إلى ملف التسديد التاريخى (PayHist.dbf) كما يتم مسحها من ملف حركة التسديد (Payments.dbf).

٥ - بعد الإنتهاء من ترحيل سجلات الفواتير التى تم تسديدها يتم تعديل محتويات حقل الموازنة الإبتدائية للشهر (Starting Balance) وكذلك حقل تاريخ آخر تحديث (Last_Updat) باستخدام الأمر (REPLACE).

٦ - يقوم البرنامج أيضا باستخدام كلمة مرور (Password) لتأمين عملية ترحيل البيانات حتى لاتتم بواسطة شخص غير مسئول. وقد تم استخدام كلمة (Mohamed) ككلمة مرور ولكن يمكن استخدام أى كلمة أخرى. ونظرا لأن عملية

الترحيل الشهرى لحركة الصرف والتسديد قد تأخذ وقتا طويلا إذا كان الملف كبيرا جدا لذلك يتم كتابة السطور التالية :

```
*     - - - -Show progress
@ 20,1 SAY "Aging the balances : Record" + ;
  STR(RECNO(),4) + "OF" + STR(RECCOUNT() , 4)
```

وهذه السطور عند تنفيذها فى البرنامج تؤدى إلى ظهور الآتى على الشاشة :

```
Aging the balance : Record 2 of 100
```

ومع نقل كل سجل يزيد العدد التالى لكلمة (Record) بواحد حتى إنتهاء عملية النقل. وهذه الرسالة تؤدى إلى اطمئنان المستخدم أن عملية النقل تتم بدون مشاكل.

والبرنامج (Post.prg) يتم كتابته كالآتى :

```
* * * * * * * * * * * * * * * * * * * * * *  Post.prg
*    Posts summarized monthly accounts to the Customer
*    file. Called from AR main menu .
SET DELETED ON
CLEAR
DO Title WITH "Monthly Posting"
* - - - - -  Display and get password
TEXT

        This is the program to post Payments and charges.
        Be sure you have printed all the monthly invoices
        before proceeding with this program.

ENDTEXT
Password = SPACE(7)
@ 15,12 SAY "Enter password to proceed" GET Password ;
     PICT "!!!!!!!"
```

```
READ

* - - - - If proper password not entered return to the
* - - - - - -  menu.
IF Password # "MOHAMED"
        ? "Illegal password" , CHR(7)
        SET DELETED OFF
        RETURN
ENDIF

* - - Do the posting,first,shift all current 30 , 60
* - - and 90 day billings "back" one field in the customer file.
@ 20,1 SAY "Working ...."
CLOSE DATABASES
USE Customer
REPLACE ALL ;
   Bal_90Plus WITH Bal_90Plus + Bal_90, ;
   Bal_90 WITH BAL_60, ;
   BAL_60 WITH Bal_30, ;
   Bal_30 WIHT Chg_Curr - Pay_Curr
REPLACE ALL ;
   Chg_Curr WITH 0 , ;
   Pay_Curr WITH 0

* - - - Now,create summary of the charges database by customer number.
SET SAFETY OFF
USE Charges INDEX ChrgNo
COPY STRUCTURE TO Summary
TOTAL ON Cust_No TO Summary FIELDS Qty, ;
   Unit_Price ,Amount FOR Billed
* - -Now update the customer database current balances
* - -with data from the charges summary file.
```

```
SELECT 1
USE Customer INDEX CustNo
SELECT 2
USE Summary
SELECT 1
UPDATE ON Cust_No FROM Summary REPLACE Chg_Curr WITH ;
  B -> Amount

* - - - Move all posted transactions to the billing
* - - history file
SELECT 2
USE BillHist
APPEND FROM Charges FOR Billed

* - - - - -  Then empty the current charges file.
CLOSE DATABASE
USE Charges INDEX ChrgNo
DELETE ALL FOR Billed
PACK

* - - Now , summarize payment totals for each Customer
USE Payments INDEX PayNo
COPY STRUCTURE TO Summary
TOTAL ON Cust_No TO Summary Fields Amount FOR Posted

* - -Now update the customer database current balances
* - -with data from the payments summary file.
SELECT 1
USE Customer INDEX CustNo
SELECT 2
USE Summary
SELECT 1
```

```
UPDATE ON Cust_No FROM Summary REPLACE Pay_Curr ;
  WITH  B -> Amount

* - - Append all posted transactions to the payments history file.
SELECT 2
USE PayHist
APPEND FROM Payments FOR Posted

* - - - - - Then empty the current payments file.
CLOSE DATABASES
USE Payments INDEX PayNo
DELETE ALL FOR Posted
PACK

* - - - Then update the 'last billed' and 'starting
* - - balance' Fields in the customer database.
USE Customer
REPLACE ALL Start_Bal WITH Start_Bal + Chg_Curr - Pay_Curr
REPLACE ALL Last_Updat WITH DATE()

* - - - - - Adjust aged balances.
GO TOP
DO WHILE .NOT. EOF()
        * - - - - - Show progress.
        @ 20,1 SAY "Aging the balances : Record" + ;
              STR(RECNO(), 4) + "OF" + STR(RECCOUNT(), 4)
        * - - - - - IF no payment, skip calculations.
        IF Pay_Curr < = 0
                SKIP
                LOOP
        ENDIF (Pay_Curr < = 0)
        * - - - - - Otherwise , subtract the payment
```

```
More = .F.
NextBal = .T.
IF Bal_90Plus > 0
        Remain = Pay_Curr - Bal_90Plus
        IF Remain > = 0
                REPLACE Bal_90Plus WITH 0
                More = .T.
        ELSE
                REPLACE Bal_90Plus WITH ABS(Remain)
                NextBal = .F.
        ENDIF(Remain > = 0)
ENDIF(90+ > 0)
* - - - - - 90 days.
IF NextBal .AND. Bal_90 > 0
        IF More
                Remain = Remain - Bal_90
        ELSE
                Remain = Pay_Curr - Bal_90
        ENDIF(More)
        IF Remain > = 0
                REPLACE Bal_90 WITH 0
                More = .T.
        ELSE
                REPLACE Bal_90 WITH ABS(Remain)
                NextBal = .F.
        ENDIF(Remain >= 0)
ENDIF (NextBal & Bal90 > 0)

* - - - - - 60 days.
IF NextBal .AND. Bal_60 > 0
        IF More
                Remain = Remain - Bal_60
```

```
		ELSE
			Remain = Pay_Curr - Bal_60
		ENDIF(More)
		IF Remain > = 0
			REPLACE Bal_60 WITH 0
			More = .T.
		ELSE
			REPLACE Bal_60 WITH ABS(Remain)
			NextBal = .F.
		ENDIF(Remain >= 0)
	ENDIF (NextBal & Bal60 > 0)

	* - - - - - 30 days.
	IF NextBal .AND. Bal_30 > 0
		IF More
			Remain = Remain - Bal_30
		ELSE
			Remain = Pay_Curr - Bal_30
		ENDIF(More)
		IF Remain > = 0
			REPLACE Bal_30 WITH 0
			More = .T.
		ELSE
			REPLACE Bal_30 WITH ABS(Remain)
			NextBal = .F.
		ENDIF(Remain >= 0)
	ENDIF (NextBal & Bal30 > 0)
	SKIP
ENDDO (end of file)

* - - - - - Display closing messages
CLEAR
```

```
? CHR(7)
TEXT

        The posting procedure is complete. Use option 4,
         from the main menu to print current monthly
        summary and aging reports.

ENDTEXT

* - - - - - Get rid of any old keypresses.
CLEAR TYPEAHEAD
WAIT "Press any key to return to main menu ...."
* - - - - - Return to the main menu.
SET DELETED OFF
CLOSE DATABASES
RETURN
```

ويلاحظ فى البرنامج استخدام عدة أوامر (IF) فى تنفيذ الحسابات الخاصة بكل حالة. فمثلا إذا لم يتم تسديد أى قيمة جديدة أى (Pay_Curr <= 0) فإن هذا يعنى أنه ليست هناك حسابات مطلوب إجراؤها ولذلك يتم الإنتقال إلى العميل التالى. وخلاف ذلك يتم ضبط حسابات (٣٠) يوما و (٦٠) يوما و (٩٠) يوما وأكثر من ٩٠ يوما لكل عميل.

ويتم استخدام المتغير (Remain) فى عملية الضبط. فإذا كان المبلغ الذى تم تسديده يغطى الحسابات القديمة أكثر من ٩٠ يوما فى هذه الحالة يتم تصفير هذه الحسابات (جعلها تساوى صفرا). وهكذا يتم المرور على باقى الحسابات (٩٠) يوما و (٦٠) يوما و (٣٠) يوما على الترتيب.

وعند الإنتهاء من حساب جميع العملاء يتم عرض رسالة للمستخدم عن التقارير المطلوبة ثم يتم إغلاق جميع الملفات والعودة إلى القائمة الرئيسية.

الباب الثالث والعشرون

برنامج التكامل بين حسابات العملاء والمخازن

يستخدم هذا البرنامج فى ربط برنامج حسابات العملاء ببرنامج المخازن. ولتنفيذ ذلك يجب إجراء التعديلات التالية :

١ - يتم طرح كميات الأصناف التى تم إدخالها فى حركة الصرف من كميات الأصناف الموجودة فى ملف المخازن الرئيسى (Master.dbf).
٢ - يتم تمييز السجلات التى يتم ترحيلها إلى ملف المخازن الرئيسى (Master.dbf) بعلامة معينة حتى لايتم ترحيلها مرة أخرى.
٣ - يتم اختبار حركة الصرف (Charges) فى برنامج حسابات العملاء أو حركة السداد (Payemnts) قبل ترحيلها إلى الملف الرئيسى للمخازن.

وفى البداية يجب إضافة حقل جديد إلى ملف حركة الصرف (Charges.dbf) يوضح للبرنامج إذا كان السجل قد تم ترحيله إلى ملف المخازن أم لا. وهذا الحقل يكون حقلا منطقيا ونسميه (InPost) حيث يمثل الحرفان (In) أول حرفين فى كلمة (Inventory) ويتم ذلك باستخدام الأمر (MODIFY STRUCTURE). كما يتم إضافة نفس الحقل إلى ملف الصرف التاريخى (BillHist.dbf) حتى يصبح بنفس التركيب. حيث أن هذا الملف يستقبل السجلات التى يتم ترحيلها إلى ملف حسابات العملاء الرئيسى(Customer.dbf).

ولتأمين عملية الترحيل يجب التأكد أن كل سجل جديد يتم إضافته إلى ملف حركة الصرف (Charges.dbf) يتم تمييزه حتى يعلم البرنامج أنه لم يتم ترحيله إلى ملف المخازن الرئيسى (Master.dbf). ويتم ذلك عن طريق إضافة سطر معين إلى برنامج إضافة العملاء (NewChrg.prg) فى الجزء الخاص بإضافة السجل الجديد. حيث يتم إضافة السطر التالى :

```
REPLACE InPost WITH .F.
```

ويصبح هذا الجزء من البرنامج كالآتى :

```
IF .NOT. Exiting
      SELECT 2
      APPEND BLANK
      REPLACE Cust_No WITH M_Cust_No
```

```
        REPLACE Date WITH DATE()
        REPLACE Billed WITH .F.
        REPLACE InPost WITH .F.
        SET FORMAT TO FNewChrg
        READ
        CLOSE FORMAT
        REPLACE Amount WITH Qty * Unit_Price
ENDIF
```

ثم يتم كتابة البرنامج الذى يقوم بتحديث ملف المخازن الرئيسى من كل السجلات التى لم يتم ترحيلها (InPost=.F.) من ملف حركة الصرف (Charges.dbf) وملف الصرف التاريخى (BilHist.dbf) كما يقوم بتغيير حقل التحديث (Inpost) إلى (True) حتى لايتم ترحيله مرة ثانية.

وحيث أن البرنامج (Updater.prg) هو البرنامج الذى يقوم بتحديث ملف المخازن الرئيسى. لذلك يتم إضافة سطر إلى هذا البرنامج يؤدى إلى تشغيل برنامج آخر إسمه (ARUpdate.prg) يختص بتحديث الملف الرئيسى من ملف حركة الصرف. ويتم إضافة هذا السطر قبل نهاية ملف التحديث (Updater) كالآتى مثلا :

```
DO ARUpdate
```

والبرنامج (ARUpdate.prg) يتكون من السطور التالية :

```
* * * * * * * * * * * * * * * * * * * * *ARUpdate.prg
*   Update the master file from charges and BilHist.
CLEAR
? "Updating from the A/R System .."
* - - - - -  Use the charges databases.
USE Charges
* - - - - -  Copy nonupdated records to Temp File.
COPY STRUCTURE TO Temp
COPY TO Temp FOR .NOT. InPost
```

```
* - - - - -  Now get the recrods from BillHist.
USE Temp
APPEND FROM BillHist FOR .NOT. InPost
* - - - - -  Get the Temp file sorted by part number.
INDEX ON Part_No TO ARIndex

* - - - - -  Use the Master file for updating .
SELECT 1
USE Master INDEX Master
SELECT 2
USE Temp INDEX ARIndex

* - - - Update Master from the temporary A/R file.
SELECT 1
UPDATE ON Part_No FROM Temp REPLACE Qty WITH ;
   Qty - Temp -> Qty

* - - - Use the origingal charges database file
* - - - change all posted fields to true
CLOSE DATABASES
USE Charges
REPLACE ALL InPost WITH .T.

* - - - - - -  Do the same thing with BillHist file.
USE BillHist
REPLACE ALL InPost WITH .T.
* - - - - -  Return to the updater command file.
RETURN
```

5

الجزء الخامس

بعـــض الأدوات المتقـدمــة

مقـدمـة

هذا الجزء يقدم مجموعة من الأدوات المتقدمة (Advanced Tools) التى يستطيع مخطط البرامج استخدامها فى كثير من التطبيقات. وهذه الأدوات بالإضافة الى ماتوفره للمستخدم من جهد ووقت فإنها أيضا تزيد من كفاءة البرنامج الذى يجرى إعداده.

ويتكون هذا الجزء من ثلاثة فصول الفصل الأول يشرح برنامج يمكن المستخدم من طباعة الشيكات مع القدرة على كتابة مبلغ الشيك بالأرقام و الحروف. و الفصل الثانى يتيح لمخطط البرامج استخدام الألوان فى الشاشات و القوائم التى يتم عرضها على المستخدم. كما يتيح للمستخدم إختيار الألوان المناسبة له من خلال قائمة إختيارت خاصة. والفصل الثالث يتيح لمخطط البرامج تصميم قوائم اختيارات تسمح للمستخدم بتحريك مؤشر على الشاشة إلى الإختيار المطلوب و ذلك علاوة على الطريقة التقليدية فى الإختيار عن طريق الأرقام أو الحروف.

الفصل الرابع والعشرون

برنامج كتابة الشيكات

فى معظم الأحيان يحتاج مخطط البرامج إلى طباعة الشيكات من خلال البرنامج. وعادة ما يحتوى الشيك على الرقم الممثل للمبلغ المطلوب صرفه كما يحتوى أيضا على ترجمة إنجليزية لهذا الرقم. فمثلا الرقم (950) يتم كتابته كالآتى :

Nine Hundred and Fifty

ولكن الأعداد التى يتم الحصول عليها من برنامج قاعدة البيانات تكون دائما على الصورة العددية.

وحيث أن البرنامج لايعرف مقدما القيمة العددية التى يتم كتابتها فى الشيك لذلك يصبح فى منتهى الصعوبة على مخطط البرامج تحويل كل عدد إلى الصورة الحرفية عند كتابته فى الشيك. ولذلك فإن هذا البرنامج يتيح لمخطط البرامج ترجمة كل عدد من (1) حتى (999,999.99) إلى الأعداد الحرفية المقابلة.

ولتوضيح ذلك يتم أولا إنشاء ملف قاعدة بيانات لكتابة الشيك حتى يتم عن طريقه اختبار البرنامج بعد ذلك. ويتكون هذا الملف من الحقول التالية :

Field	Field Name	Type	Width	Dec
1	CHECK_NO	Numeric	5	0
2	TO_WHOM	Character	25	
3	AMOUNT	Numeric	9	2
4	DATE	Date	8	
TOTAL			**48**	

شكل (٢٤ - ١)

وقبل كتابة البرنامج يجب أولا تخزين المقابل الحرفى لكل عدد من الأعداد من واحد إلى عشرين فى متغيرات ذاكرة وكذلك أرقام العشرات مثل (Thirty) و (Fourty) و .... الخ كما يتم تخزين هذه المتغيرات فى ملف ذاكرة (Memory File). وهذا الملف نسميه (English.mem) ويتم إنشاؤه من خلال البرنامج التالى :

```
* * * * * * * * * * * * * * * * * *English.prg
* sets up memory file for storing English equivalents.
CLEAR
? "Creating English.mem file with English for numbers"
?
SET DEFAULT TO C
SET TALK ON
CLEAR MEMORY
U  = " "
U1 = "ONE"
U2 = "TWO"
U3 = "TREE "
U4 = "FOUR "
U5 = "FIVE "
U6 = "SIX "
U7 = "SEVEN "
U8 = "EIGHT "
U9 = "NINE "
U10 = "TEN "
U11 = "ELEVEN "
U12 = "TWELVE "
U13 = "THIRTEEN "
U14 = "FOURTEEN "
U15 = "FIFTEEN "
U16 = "SIXTEEN "
```

```
U17 = "SEVENTEEN"
U18 = "EIGHTEEN"
U19 = "NINETEEN"
U20 = "TWENTY"
U30 = "THIRTY"
U40 = "FOURTY"
U50 = "FIFTY"
U60 = "SIXTY"
U70 = "SEVENTY"
U80 = "EIGHTY"
U90 = "NINETY"

* - - - - - Save all variables to English.mem file .
SAVE TO English
CLEAR
?
?
?
SET TALK OFF
RETURN
```

وهذا البرنامج يؤدى إلى إنشاء ملف الذاكرة (English.mem) الذى يجب تحميله فى الذاكرة عند تشغيل برنامج إنشاء الشيكات. ويلاحظ أن كل متغير يبدأ إسمه بالحرف (U) يليه رقم يمثل الرقم المطلوب تحويله فمثلا (U5) يمثل (Five) و (U30) يمثل (Thirty) وهكذا.

ولتحميل هذا الملف فى الذاكرة يتم كتابة السطر التالى :

```
RESTORE FROM English
```

والخطوة التالية بعد ذلك هى إنشاء ملف الخطوات (Procedure File) أو البرنامج الذى يقوم بترجمة أى عدد إلى المقابل الحرفى له. وهذا البرنامج نسميه (Translat.prg) ويتم كتابته كالآتى :

```
* * * * * * * * * * * * * * * * *Translat.prg
* - - - - -  Procedure to convert a number to English
* - - - - -  equivalent.
PROCEDURE Translat
PARAMETERS Number, English

*    Set up memory variables
Counter = 1
Start = 1
String = STR(Amount, 9, 2)
* - - - - -  Loop through thousands and hundreds
DO WHILE Counter < 3
        * - - - Split out hundreds , tens and ones.
        Chunk = SUBSTR(String, Start , 3)
        Hun = SUBSTR(Chunk , 1, 1)
        Ten = SUBSTR (Chunk, 2, 2)
        One = SUBSTR(Chunk , 3, 1)

        * - - - - -  Handle hundreds portion.
        IF VAL(Chunk) > 99
                English = English + U & Hun + "Hundreds"
        ENDIF
        * - - - - -  Handle second 2 digits.
        T = VAL(Ten)
        IF T > 0
                DO CASE
```

```
                    *---Case 1 : handle even tens and teens
                    CASE (INT(T/10.0) = T/10.0) .OR. (T > 9  ;
                        .AND. T < 20)
                        English = English + U & Ten
                    *--Case 2 : Handle greater than 10
                    *---  but not evenly divisable.
                    CASE T > 9 .AND. (INT(T/10.0) # T/10.0)
                        Ten = SUBSTR(Ten, 1,1) + '0'
                        English = English + U & Ten + " " ;
                          +U&one
                    *---Case 3 : Handle less than 10
                    CASE T < 10
                        English = English + U & One
          ENDCASE
     ENDIF (T > 0)
     *----- Add "Thousand" if necessary
     IF Amount > 999.99 .AND. Counter = 1
          English = English + "THOUSAND"
     ENDIF (need to add "Thousand")
     *----- Prepare for pass through hundreds.
     Start = 1
     Counter = Counter + 1
ENDDO (while counter < 3)
*----- Take out ratios
IF INT(Amount) > 0
     English = English + "And"
ENDIF
English = English + SUBSTR(String, 8 , 2) + "/100"
```

ويمكن استخدام هذا البرنامج داخل أى برنامج آخر عن طريق كتابة السطر التالى :

```
SET PROCEDURE TO Translate
```

ويؤدى هذا إلى فتح ملف الخطوات السابق. ويجب قبل تشغيل البرنامج تعريف المعاملات التى سوف تستخدم معه وذلك كالآتى :

```
M_Amount = 9845
English = "  "
DO Translat WITH M_Amount , English
```

ويجب قبل ذلك فتح ملف الذاكرة (English) وذلك بكتابة السطر التالى :

```
RESTORE FROM English
```

وبذلك يمكن اختبار البرنامج. ولكتابة البرنامج الذى يؤدى إلى طباعة الشيك بالكامل يتم كتابة السطور التالية :

```
* * * * * * * * * * * * * * * * *Checks.prg
* - - - - -  Sample program to write checks
CLEAR
SET TALK OFF
SET SAFETY OFF
SET PROCEDURE TO Translat
* - - - - -  Bring English equivalent variables.
RESTORE FROM English
* - - - - -  Use the checks database
USE Checks
INDEX ON Check_No TO CheckNo
?
? "First Check to be printed will be", Check_No
?
?
```

```
WAIT "Press any key to begin writing checks"
CLEAR
SET PRINT ON
GO TOP
DO WHILE .NOT. EOF()
        * - - - - - Translate Amount to English
        M_Amount = Amount
        English = "  "
        DO Translat WITH M_Amount , English

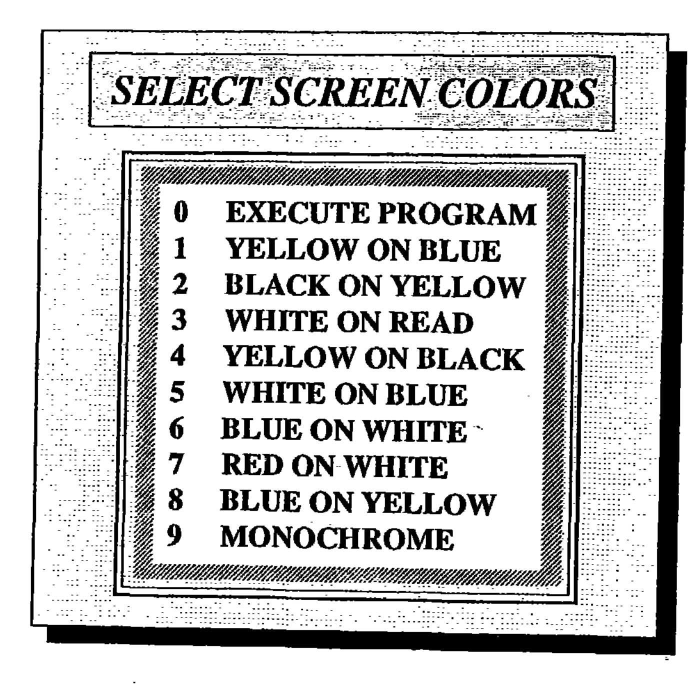
        * - - - Print the check with required modification
        * - - - - - for the check
        ? "                    " Date
        ?
        ? To_Whom, "     " , Amount
        ?
        ? English
        ?
        ?
        ?
        SKIP
ENDDO (not eof)
SET PRINT OFF
CLOSE PROCEDURE
CLEAR MEMORY
CLOSE DATABASES
RETURN
```

هذا البرنامج يؤدى إلى طباعة شيكات تحتوى على المبلغ (Amount) مكتوبا بالأرقام وكذلك بالحروف.

الفصل الخامس والعشرون

برنامج اختيار الألوان

هذا البرنامج يسمح للمستخدم باختيار ألوان الشاشة وكذلك ألوان الأعمدة الضوئية (Highlights). وذلك عن طريق عرض قائمة بالألوان المختلفة والسماح للمستخدم باختيار الألوان المطلوبة أنظر الشكل (٢٥ - ١).



شكل (٢٥ - ١)

ويمكن تسمية هذا البرنامج (Colors.prg) ويتم كتابته كالآتى :

```
* * * * * * * * * * * * * * * * * *Colors.prg
* - - - - -  to give the user the colors he needs.
CLEAR
SET TALK OFF
SET STATUS OFF
* - - - - -  Set up loop for displaying the menu
DO WHILE .T.
        * - - - - -  Drow a box
        @ 1,24 TO 3,50 DOUBLE
```

```
* - Set up loop for drawing a frame inside the box
R = 6

DO WHILE R > 5 .AND. R < 20
       @ R,22 SAY REPLICATE(CHR(178),31)
       R = R + 1
ENDDO

* - - - - - Erase an area from the box
@ 7, 24 SAY CLEAR TO 18,50
@ 5, 21 TO 20,53 DOUBLE

* - - - - - Display the menu."
@ 2, 28 SAY "Select Screen Colors"
@ 8, 25 SAY " 0 - Execute Program"
@ 9, 25 SAY " 1 - Yellow On Blue"
@ 10,25 SAY " 2 - Black On Yellow"
@ 11,25 SAY " 3 - White On Red"
@ 12,25 SAY " 4 - Yellow On Black"
@ 13,25 SAY " 5 - White On Blue"
@ 14,25 SAY " 6 - Blue On White"
@ 15,25 SAY " 7 - Red On White"
@ 16,25 SAY " 8 - Blue On Yellow"
@ 17,25 SAY " 9 - Monochrom
WAIT "    " TO P
DO CASE
       CASE P = 0
              RETURN
       CASE P = 1
              COLSTR = "GR + /B, W/R, GR"
       CASE P = 2
              COLSTR = "N/GR, W/R, GR"
```

```
        CASE P = 3
            COLSTR = "W/R , W/N"
        CASE P = 4
            COLSTR = " GR + /N , W/R , GR"
        CASE P = 5
            COLSTR = " W/B , W/R , B"
        CASE P = 6
            COLSTR = " B/W , W/R , R"
        CASE P = 7
            COLSTR = "R/W , W/N , B"
        CASE P = 8
            COLSTR = "B/GR , W/R , B"
        CASE P = 9
            COLSTR = "7/0 , 0/7 , 0"
    ENDCASE
    SET COLOR TO & COLSTR
CLEAR
ENDDO
```

ويلاحظ فى بداية البرنامج استخدام حلقة تكرارية لرسم مستطيل داخلى وملؤه بالتظليل باستخدام الحرف CHR(178) ثم استخدام الأمر (CLEAR) بعد ذلك لمسح الجزء الداخلى من هذا المستطيل حتى يظهر كإطار حول قائمة الإختيارات.

كما يلاحظ أن الألوان التى تظهر على الشاشة (COLSTR) تعتمد على الرقم المخزن فى متغير الذاكرة (P) والذى تم اختياره من قائمة الألوان. وهذا البرنامج يمكن استخدامه مع أى برنامج آخر بكتابة السطر التالى :

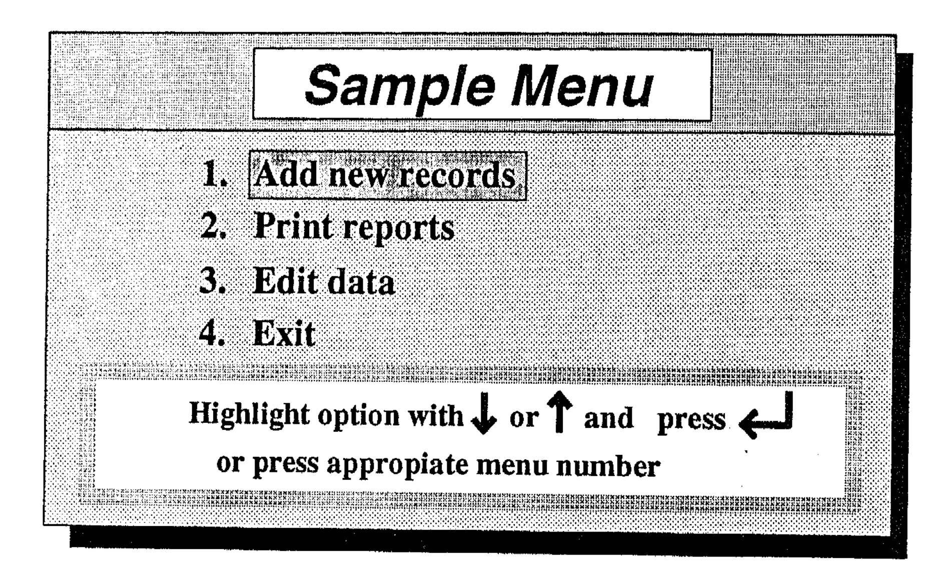
```
DO Colors
```

الفصل السادس والعشرون

برنامج تحريك العمود الضوئى

هذا البرنامج يستخدم عندما يريد مخطط البرامج إضافة مزيد من الإثارة والتشويق إلى قوائم الإختيارات. حيث يمكن عند تنفيذه عرض عمود ضوئى (Highlight) يمكن تحريكه بواسطة مفاتيح الإتجاهات لأعلى ولأسفل للوصول إلى الإختيار المطلوب ثم الضغط على مفتاح الإدخال لتنفيذ هذا الإختيار. وفى نفس الوقت يتيح للمستخدم الإختيار عن طريق كتابة الرقم الموجود عند أى اختيار.

وعند تنفيذ هذا البرنامج تظهر الشاشة التالية :



شكل (٢٦ - ١)

والسطور التالية توضح نموذجا لبرنامج (Sample.prg) يستخدم هذه الطريقة فى عرض القائمة الرئيسية :

```
* * * * * * * * * * * * * * * * *Sample.prg
*   Create a sample menu with moving "light bar".
CLEAR
SET TALK OFF
```

```
* - - - - -  create menu options (opt1-opt4)
opt1 = "1. Add new records"
Opt2 = "2. Print reports"
Opt3 = "3. Edit data"
Opt4 = "4. Exit"

* - - - - -  Display the menu
@ 1,1  TO 3,79 DOUBLE
@ 2,32 SAY "Sample menu"
@ 5,30 SAY Opt1
@ 6,30 SAY Opt2
@ 7,30 SAY Opt3
@ 8,30 SAY Opt4

* - - Display instructions with graphics characters.
@ 14,1 TO 18,78 DOUBLE
@ 15,18 SAY "Highlight option by using"
@ 15,40 SAY CHR(24) + "or" + CHR(25) + "and press" ;
   + CHR(17) + CHR(217)
@ 17,22 SAY "or press appropriate menu number"

* - - - - -  Initialize memory variables.
Opt = 1
Sub = STR(Opt,1)
KeyPress = 0
Choice = 0
* - - - - -  Reverse video on option 1
@ 5,30 GET Opt1
CLEAR GETS

* - - - - -  Loop for choosing menu options.
DO WHILE Choice # 4
```

```
*----- Wait for a keyPress.
KeyPress = 0
DO WHILE KeyPress = 0
    KeyPress = INKEY()
ENDDO (KeyPress)

*----- Arrow key pressed.
IF KeyPress = 24 .OR. KeyPress = 25
    @ Opt+4,30 SAY Opt&Sub
    Opt = IIF(KeyPress = 24, Opt+1, Opt - 1)
    Opt = IIF(Opt > 4 , 1, Opt)
    Opt = IIF(Opt < 1 , 4, Opt)
    Sub = STR(Opt,1)
    @ Opt+4,30 GET Opt &Sub
    CLEAR GETS
    LOOP
ENDIF

*----- Option numbers entered.
IF KeyPress >= 49 .AND. KeyPress <= 52
    Choice = KeyPress - 48
ENDIF

*----- Return pressed
IF KeyPress = 13
    Choice = opt
ENDIF
*----- An option was selected.
IF Choice > 0
    CLEAR
    ? 'Choice' , Choice
    Choice = 5
ENDIF
```

```
ENDDO(Choice)
```

والبرنامج يبدأ بتخزين كل سطر من سطور القائمة فى متغير ذاكرة وذلك يتضح من السطور التالية :

```
* - - - - - create menu options (opt1-opt4)
opt1 = "1. Add new records"
Opt2 = "2. Print reports"
Opt3 = "3. Edit data"
Opt4 = "4. Exit"
```

والجزء التالى من البرنامج يؤدى إلى ظهور قائمة الاختيارات على الشاشة ويتكون من السطور التالية :

```
* - - - - Display the menu
@ 1,1  TO 3,79 DOUBLE
@ 2,32 SAY "Sample menu"
@ 5,30 SAY Opt1
@ 6,30 SAY Opt2
@ 7,30 SAY Opt3
@ 8,30 SAY Opt4
```

والجزء التالى من البرنامج يتم عن طريقه عرض لوحة المساعدة (Help) التى تظهر على الشاشة لتوضح للمستخدم كيفية الإختيار من القائمة. ويلاحظ استخدام حروف الرسم (Graphic Characters) التى يتم الحصول عليها باستخدام الدالة (CHR). حيث يتم كتابة السهم العلوى (↑) باستخدام الدالة CHR(24) ويتم كتابة السهم لأسفل باستخدام الدالة CHR(25). ويتم استخدام الدالة CHR(17) فى كتابة حروف رسم معينة تمثل شكل مفتاح الإدخال (↵) وهكذا. ويتضح ذلك من السطور التالية :

```
@ 14,1 TO 18,78 DOUBLE
@ 15,18 SAY "Highlight option by using"
@ 15,40 SAY CHR(24) + "or" + CHR(25) + "and press" ;
```

```
+ CHR(17) + CHR(217)
@ 17,22 SAY "or press appropriate menu number"
```

وفى الجزء التالى يتم إنشاء مجموعة من متغيرات الذاكرة مثل المتغير (Opt) الذى يتم فيه تخزين الرقم الذى يتم اختياره والمتغير (Sub) الذى يتم فيه تخزين القيمة الحرفية لهذا الرقم ، والمتغير (KeyPress) الذى يتم فيه تخزين كود الآسكى الخاص بالمفتاح الذى يضغط عليه المستخدم والمتغير (Choice) الذى يتم فيه تخزين الإختيار المطلوب لاستخدامه بعد ذلك فى التفرع إلى البرنامج الخاص به. ويتم ذلك من خلال السطور التالية :

```
Opt = 1
Sub = STR(Opt,1)
KeyPress = 0
Choice = 0
```

والجزء التالى من البرنامج يؤدى إلى ظهور عمود ضوئى فى المكان الممثل للإختيار رقم (1) فى القائمة وذلك باستخدام الأمر (GET). كما يتم مسح المتغير (GET) الذى يتكون نتيجة لذلك مع بقاء العمود الضوئى على هذا الإختيار. ويتم ذلك من خلال السطرين التاليين :

```
@ 5,30 GET Opt1
CLEAR GETS
```

والجزء التالى يتم من خلاله تكوين حلقة تكرارية لعرض القائمة دائما على الشاشة حتى يختار المستخدم الخروج. وذلك من خلال السطر التالى :

```
DO WHILE Choice # 4
```

كما تستخدم حلقة تكرارية أخرى لانتظار ضغط المستخدم على أى مفتاح. وفى هذه الحالة يتم تخزين كود الآسكى (ASCII Code) الخاص بهذا المفتاح فى متغير الذاكرة (KeyPress). وتستخدم الدالة INKEY() فى الحصول على كود الآسكى الخاص بآخر مفتاح قام المستخدم بالضغط عليه. ويتضح ذلك من السطور التالية :

```
KeyPress = 0
DO WHILE KeyPress = 0
       KeyPress = INKEY()
ENDDO (KeyPress)
```

وعند ضغط المستخدم على مفتاح السهم لأعلى (CHR(24)) أو مفتاح السهم لأسفل (CHR(25)) فإن ذلك يؤدى إلى إختفاء العمود الضوئى من الإختيار رقم (١). وذلك لأن الأمر (SAY) يؤدى إلى إعادة كتابة السطر فوق العمود الضوئى كما يؤدى إلى إضافة واحد أو طرح واحد من الرقم المخزن فى المتغير (Opt). ويلاحظ هنا استخدام الدالة (IIF) فى زيادة قيمة المتغير (Opt) أو إنقاصه حسب القيمة المخزنة فى المتغير (KeyPress) نتيجة ضغط المستخدم على مفتاح معين. ويتضح ذلك من السطور التالية :

```
IF KeyPress = 24 .OR. KeyPress = 25
       @ Opt+4,30 SAY Opt&Sub
       Opt = IIF(KeyPress = 24, Opt+1, Opt - 1)
       Opt = IIF(Opt > 4 , 1, Opt)
       Opt = IIF(Opt < 1 , 4, Opt)
       Sub = STR(Opt,1)
       @ Opt+4, 30 GET Opt&Sub
       CLEAR GETS
       LOOP
ENDIF
```

وعندما يريد المستخدم الإختيار بالرقم وليس عن طريق تحريك العمود الضوئى فإنه يكتب رقما بين (1) و (4). وحيث أن كود الآسكى الخاص بالأرقام يبدأ من (48) لذلك فإن قيمة العدد الذى يختاره المستخدم يمكن حسابها بطرح العدد (48) من العدد المثل لكود الآسكى (ASSII) الخاص بهذا العدد. ويتم ذلك من خلال السطور التالية :

```
* - - - - Option numbers entered.
IF KeyPress > = 49 .AND. KeyPress < = 52
       Choice = KeyPress - 48
ENDIF
```

وعندما يضغط المستخدم على مفتاح الإدخال (CHR(13)) يتم تخزين العدد الموجود فى المتغير (Opt) فى المتغير (Choice). ويتم ذلك من خلال السطور التالية :

```
IF KeyPress = 13
        Choice = opt
ENDIF
```

وسواء كتب المستخدم رقم الإختيار المطلوب أو استخدم مفاتيح الإتجاهات فى تحريك العمود الضوئى ثم ضغط على مفتاح الإدخال فإن المتغير (Choice) يتم فيه تخزين رقم معين يمثل هذا الإختيار.

وفى برنامج القائمة العادية يتم استخدام الأمر (DO CASE) فى التفرع إلى برنامج معين بناء على الرقم الموجود فى المتغير (Choice). أما فى هذا البرنامج فقد تم الإكتفاء بعرض الرقم الموجود فى المتغير (Choice) حتى يتم اختبار البرنامج والتأكد من تحقيقه للمطلوب.



الجزء السادس

تطبيقـــات إضـافيـــة

فى الأجزاء السابقة من الكتاب تم شرح مجموعة التطبيقات المحاسبية الشائعة التى تهم رجال الأعمال. كما تم شرح بعض الوسائل المتقدمة التى يمكن لمخطط البرامج استخدامها فى أى برنامج لزيادة كفاءته. وقد روعى فيما سبق أن يتدرج الكتاب فى درجة صعوبة البرامج حتى يصل بالقارىء فى نهاية الكتاب إلى الخبرة الكافية والقدرة على التعامل مع أعقد نظم المعلومات. ولكن عند طباعة هذه النسخة الجديدة من الكتاب فقد رأينا إضافة تطبيق آخر يتميز بالسهولة والبساطة وفى نفس الوقت يستطيع المستخدم الذى يريد الإستفادة بالحاسب فى المنزل تطبيقه وتحقيق فائدة عملية من استخدامه.

والتطبيق الجديد ببساطة هو برنامج يتيح للمستخدم تخزين بيانات المعارف والأقارب والأصدقاء متضمنة الإسم والعنوان ورقم التليفون وأى ملاحظات أخرى كما يتيح له إسترجاع أى بيانات عن أى شخص بسهولة وكذلك طباعة دليل (Directory) يتضمن بيانات أى شخص أو مجموعة من الأشخاص أو طباعة عناوين بريدية (Lables) تعطى بيانات مختصرة عن هؤلاء الأشخاص. وبالإضافة إلى ذلك فإن البرنامج يتيح للمستخدم تعديل بيانات الأصدقاء أو مسح أى بيانات أو إضافة بيانات جديدة وذلك من خلال قوائم واضحة تسهل على المستخدم المبتدىء تشغيل البرنامج دون الحاجة إلى أى معلومات عن قواعد البيانات.

ويتضمن هذا الجزء بالإضافة إلى التطبيق المذكور شرح مولد التطبيقات (Application Generator) الخاص ببرنامج (+ DBase III) مع شرح أحد التطبيقات التى يمكن تنفيذها من خلاله.

الفصل السابع والعشرون

التطبيــق المــنزلى

يتكون هذا النظام من أربعة برامج يتم تشغيلها من خلال الإختيارات الموجودة فى برنامج خامس. وهذا البرنامج الخامس هو البرنامج المحتوى على قائمة الإختيارات الرئيسية الخاصة بالنظام وهى القائمة التى يراها المستخدم فى بداية تشغيل البرنامج ويتم من خلالها تشغيل كل برنامج من البرامج الأربعة عند اختيار المستخدم له. وتعود القائمة للظهور عند الإنتهاء من البرنامج حتى يستطيع المستخدم تشغيل برنامج آخر أو الخروج من النظام. ويتم تشغيل برنامج القائمة الرئيسية بكتابة الأمر التالى :

```
DO HOME
```

وهذا يؤدى إلى عرض القائمة التالية على الشاشة :

```
   Home System Main Menu

1 - Add new Names and Addresses
2 - Print Directory or Labels
3 - Make Changes
4 - Delete Names and Addresses
5 - Check for Duplicate Entries
6 - Exit the Home system
    Enter choice :
```

وسوف نوضح فى الجزء التالى ما يحدث عند اختيار المستخدم لكل من هذه الإختيارات.

أ - إضافة أسماء جديدة

لإضافة أسماء وعناوين جديدة إلى قاعدة البيانات يختار المستخدم الإختيار رقم (١) من القائمة الرئيسية. والشكل (٢٧ - ١) يوضح شاشة إدخال البيانات التى تظهر فى هذه الحالة. وبعد إدخال البيانات تعود القائمة الرئيسية للظهور مرة أخرى.

Enter names and addresses

Name : ▒▒▒▒▒ City : ▒▒▒▒▒

Address : ▒▒▒▒▒

Phone : ▒▒▒▒▒ Company : ▒▒▒▒▒

شكل (٢٧ - ١)

ب - طباعة العناوين البريدية (Labels) أو الدليل (Directory)

لتنفيذ ذلك يختار المستخدم الإختيار رقم (٢) فى القائمة وفى هذه الحالة تظهر قائمة فرعية كالآتى :

Select a Report Option

1 - Directory
2 - Mailing Lables
3 - Return to Main Menu
Enter your choice

وعند اختيار المستخدم لأى من هذه الإختيارات ما عدا الإختيار رقم (٣) تظهر قائمة فرعية كالآتى :

Select a Sort Order

1 - Alphabetical order by name
2 - City order
3 - Original order

وبعد اختيار المستخدم للترتيب المطلوب يظهر السؤال التالى :

Do you want (A)ll records or (Q)uery?

واختيار (All) يتم عن طريق كتابة الحرف (A) صغيرا أو كبيرا ويؤدى إلى عرض كل السجلات فى قاعدة البيانات. واختيار (Query) يتم عن طريق كتابة الحرف (Q) ويؤدى إلى عرض نموذج البحث (Query Form) المبين بالشكل (٢٧ - ٢) والذى يتم من خلاله إدخال شروط البحث. وعندما يملأ المستخدم نموذج البحث الخاص بسجلات معينة ويختار (Exit) ثم (Save) من القائمة فإن البرنامج يسأل

Send report to printer? (Y / N).

```
Set Filter                    Nest                    Display

Operator                 Begins with
Constant/Expression      "A"
Connect

Line Number              1
```

Line	Field	Operator	Constant/Expression
1	NAME	Begins with	"A"
2			
3			
4			
5			

شكل (٢٧ - ٢)

وينتظر اجابة المستخدم. فعندما يكتب (Y) يتم إرسال التقرير إلى الطابعة وعندما يكتب (N) يتم عرضه على الشاشة فقط. والشكل (٢٧ - ٣) يوضح عينة (Sample) للدليل (Directory) المطبوع بواسطة النظام. كما يوضح الشكل (٢٧ - ٤) تقرير العناوين البريدية (Mailing Labels) المطبوع بواسطته.

```
Page No.      1
07/07/93
                         Home System Directory
ASHRAF MAHMOUD          ALAMANA COMPANY          765787
                        23-OMAR LOTFY            CAIRO
FAWZY FAHIM HASAN       ALSHOROK COMPANY         7656787
                        12- ALREAD STREET        ASWAN
HASAN MASOUD LOTFY      ELZIOT COMPANY           675778
                        54-SHBRA STREET          CAIRO
HASN FOAAD              ELNAHAR COMPANY          76578
                        ELTAWFIK STREET          BELBEIS
HAZEM KAMEL             GAMA                     2645321
                        16-ALTAYARAN             CAIRO
MAHMOUD GHONEIM         ELNAHAR COMPANY          4567887
                        16-MOHARAM STREET        ALEXANDRIA
SALEM AHMED ELFEKY      ELNOR COMPANY            656789
                        66-AHMED MORSY STREET    TANTA
Press any key to return to menu ..
```

شكل (٢٧ - ٣)

```
ELNAHAR COMPANY
ELTAWFIK STREET
BELBEIS

HAZEM KAMEL
GAMA
16-ALTAYARAN
CAIRO

MAHMOUD GHONEIM
ELNAHAR COMPANY
16-MOHARAM STREET
ALEXANDRIA

SALEM AHMED ELFEKY
ELNOR COMPANY
66-AHMED MORSY STREET
TANTA
   Press any key to return to menu ..
```

شكل (٢٧ - ٤)

وبعد طباعة الدليل أو العناوين البريدية تعود القائمة الرئيسية للنظام إلى الظهور.

حـ - تعديل البيانات

عندما يختار المستخدم الإختيار رقم (٣) من القائمة الرئيسية فإن البرنامج يطلب الآتى من المستخدم :

Enter name of person to Edit
or just press Return to Quit :

وعند ضغط المستخدم على مفتاح الإدخال فإنه يعود إلى القائمة الرئيسية. وإذا كتب أسماء غير موجودة فى قاعدة البيانات فإن الحاسب يطلق صفارة تحذير (Beep) ويعرض رسالة الخطأ التالية :

There is no < Name>
Press any key to try again

ملاحظة

الإسم الخطأ الذى تم إدخاله يظهر مكان < Name >.

وإذا كتب المستخدم إسما وكان هناك سجل واحد فى قاعدة البيانات يحتوى على هذا الإسم فإن النظام يسمح للمستخدم بتعديل بيانات هذا السجل من خلال الشاشة الموضحة بالشكل (٢٧ - ٥).

```
Enter or Edit Name and Address

NAME      HAZEM KAMEL                 CITY     CAIRO
ADDRESS        16-ALTAYARAN
PHONE          2645321     COMPANY  GAMA
```

شكل (٢٧ - ٥)

واذا كان هناك أكثر من سجل يحتوى على الإسم المطلوب فإن النظام يطلب معلومات أكثر عن طريق عرض بيانات السجلات المتطابقة الإسم وسؤال المستخدم عن رقم السجل المطلوب وذلك كالآتى مثلا :

304 Ashraf Ebrahim
365 Ashraf Ebrahim
713 Ashraf Ebrahim

Write the required Record Number :

وعند كتابة المستخدم لأحد أرقام السجلات والضغط على مفتاح الإدخال تظهر الشاشة الخاصة ببيانات هذا السجل. وبعد إدخال التعديلات المطلوبة تعود القائمة الرئيسية إلى الظهور.

د - مسح السجلات

يتم مسح السجلات عندما يختار المستخدم الإختيار رقم (٤) من القائمة الرئيسية. وكما ذكرنا فى تعديل البيانات فإن المستخدم يسأل عن إسم الشخص المطلوب مسح سجله ويكفى أن يكتب الإسم الأول أو الحروف الأولى منه. وإذا لم يكن هذا الإسم موجودا يطلق الحاسب صفارة تحذير (Beep) ويتيح للمستخدم المحاولة مرة ثانية. وإذا كان هناك عدة أشخاص لهم نفس الإسم فإن النظام يعرض بيانات هؤلاء الأشخاص ويطلب من المستخدم تحديد رقم السجل المطلوب. وبعد تحديد السجل المطلوب يتيح النظام للمستخدم التأكد من سلامة إختياره عن طريق عرض بيانات هذا السجل كالآتى :

Record #	Name	Address	City
1	Ashraf Mohamed	B-Abbas-ELakkad	Cairo

Delete this record? (Y / N) :

ويستطيع المستخدم كتابة (Y) أو (N) والإستمرار فى مسح السجلات الأخرى. ولإيقاف المسح فإن المستخدم يقوم ببساطة بالضغط على مفتاح الإدخال بدلا من إدخال إسم جديد. وقبل مسح السجلات فعليا من قاعدة البيانات يعرض الحاسب بيانات السجلات التى حددها المستخدم حتى يتأكد من رغبته فى مسحها وذلك كالآتى :

Records to be deleted ...

Record #	Name	Address
6	Ashraf Mohamed	13 - Abbass Elakkad
5	Medhat Salem	28 - Ain Shams

Delete all these? (Y / N) :

واذا كانت الإجابة (No) فإن النظام يتيح للمستخدم إستدعاء أحد السجلات المعروضة عن طريق كتابة رقم السجل الخاص به. وتستمر هذه العملية حتى يكتب المستخدم (Y) أمام السؤال (Delete all these?) أو عندما لايتبقى أى سجلات جاهزة للمسح. وفى هذه الحالة يتم مسح السجلات نهائيا (Pack) من قاعدة البيانات.

هـ - اختبار السجلات المكررة (Duplicates)

من المعتاد فى هذا النظام أن يتم إدخال بعض الأسماء أكثر من مرة وبالتالى تصبح هناك بعد فترة سجلات مكررة لذلك فإن النظام يتضمن إختيارا يتيح للمستخدم إكتشاف السجلات المكررة حتى يستطيع مسحها. ويتم ذلك عندما يختار المستخدم الإختيار رقم (٥) من القائمة الرئيسية للنظام. وفى هذه الحالة يسأله البرنامج إذا كان المطلوب طباعة هذه السجلات المكررة أم يكفى عرضها على الشاشة ثم يعرض تقريرا كالآتى :

Possible Duplications

Record #	Name	Address	City
31	Ayman Salah	16- Ahram street	Cairo
67	Ayman Salah	16- Ahram street	Cairo

ويجب ملاحظة أن هذا الإختيار يعرض فقط السجلات المتشابهة ولكنه لايمسحها. والقرار النهائى بمسح أى سجل يرجع إلى المستخدم. وعند إتخاذ هذا القرار فإنه يستخدم الإختيار رقم (٤) السابق توضيحه فى مسح السجلات المتكررة.

و - الخروج من النظام

يتيح الإختيار رقم (٦) للمستخدم الخروج من النظام والعودة إلى مشيرة النقطة (Dot Prompt).

وفى الأجزاء التالية من هذا الفصل يتم توضيح النظام بالتفصيل.

٢٧ - ١ تصميم قاعدة البيانات

يتم تصميم قاعدة بيانات هذا التطبيق المنزلى عن طريق كتابة الامر التالى :

```
CREATE  Homedata.dbf
```

ويتم إدخال بيانات الحقول كالآتى :

Field	Field Name	Type	Width	Dec
1	NAME	Character	25	0
2	ADDRESS	Character	25	0
3	CITY	Character	15	0
4	PHONE	Character	10	0
5	COMPANY	Character	25	0

ويمكن إنشاء ملفات الفهرس مباشرة. ولكى يتم الإحتفاظ بالسجلات مرتبة بالإسم يتم كتابة الأمر التالى :

```
INDEX ON UPPER( NAME ) TO  HOMENAME
```

وهذا الأمر يؤدى إلى إنشاء ملف الفهرس (Homename. ndx) الذى يحتوى على الاسماء مكتوبة بحروف كبيرة (Upercase). وإستخدام الدالة (Upper) يساعد على تسهيل الوصول إلى السجل فى حالة كتابة المستخدم الإسم بحروف صغيرة أو كبيرة كما سوف يتضح فيما بعد.

ويمكن فهرسة قاعدة البيانات على المدينة (City) وذلك حتى يتسنى إستدعاء بيانات الأشخاص الذين يقطنون مدينة معينة. ويتم ذلك بإستخدام الأمر التالى :

```
INDEX ON CITY + NAME To HOMCITY
```

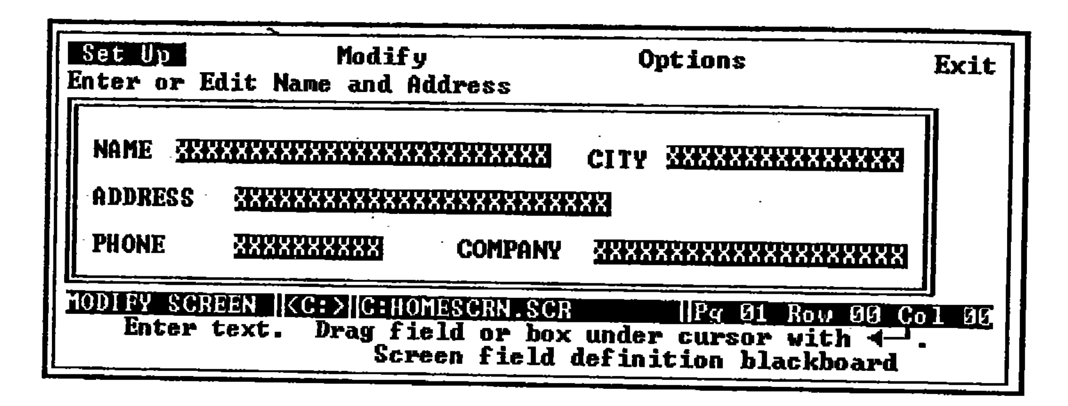
وهذا يؤدى إلى إنشاء فهرس يحتوى على أسماء الأشخاص مجمعة على المدن مع الإحتفاظ بالترتيب الهجائى (Alphabetic) داخل كل مدينة.

وعندما يراد إدخال أو تعديل أى بيانات يجب أولا فتح ملف قاعدة البيانات والملفات الفهرسية وذلك كالأتى :

USE HOMEDATA INDEX HOMENAME , HOMECITY

٢٧ - ٢ إنشاء شاشة الإدخال

شاشة الإدخال المستخدمة فى إدخال و تعديل البيانات تسمى (Homescrn). ويمكن إنشاؤها بواسطة راسم الشاشات (Screen Painter) السابق شرحه فى الأجزاء السابقة. وهى كما يتضح من الشكل (٢٧ - ٦).



شكل (٢٧ - ٦)

٢٧ - ٣ انشاء الدليل والعناوين البريدية

تصميم نموذج التقرير الخاص بالدليل (Directory) يحتاج إلى شىء من التكتيك حيث أن المطلوب هو طباعة سطرين من البيانات لكل سجل فى قاعدة البيانات. ويمكن استخدام الأمر (MODIFY) أو الأمر (CREATE) لإنشاء نمودج التقرير الذى نسميه (Newdirec.frm).

ومن خلال القائمة الأولى فى شاشة مولد التقارير يتم تحديد الهامش الأيسر بالقيمة (Zero) واختيار المسافات المزدوجة (Double Space) كما يتم إدخال عنوان التقرير كالآتى مثلا :

(Home System Directory).

ويتم إدخال العمود الأول فى التقرير مع استخدام عرض قدره ٢٥ حرفا وترك رأس العمود (Heading) خاليا وذلك بكتابة الآتى :-

TRIM (NAME)

وهذا يؤدى إلى عرض الإسم فى السطر الأول من العمود ولا خوف من نزول حروف من الإسم فى السطر التالى لأن عرض حقل الإسم لايزيد عن عرض العمود.

ويتم إدخال العمود الثانى فى نموذج التقرير كالآتى :

COMPANY + ADDRESS

ويتم تحديد عرض هذا العمود بـ (٢٥) حرفا مع ترك رأس العمود (Heading) خاليا. وحيث أن حقل (Company) عرضه (٢٥) حرفا لذلك فإن العنوان (Address) ينزل فى السطر التالى تحت (Company). وهذا يؤدى إلى عرض الحقلين فوق بعضهما كالأتى مثلا :

Delta
5. Abdulla Ebn El-zobair

والعمود الثالث يحتاج إلى تكتيك أيضا حتى يمكن ضبط حقلى رقم التليفون (Phone) والمدينة (City) فوق بعضهما تماما. ولتنفيذ ذلك يتم تحديد عرض العمود بـ (١٥) حرفا. ولكن نريد التأكد أن حقل رقم التليفون (Phone) عرضه (١٥) حرفا تماما حتى نضمن نزول المدينة (City) إلى السطر التالى. وحيث أن عرض حقل (Phone) هو (١٠) حروف فقط لذلك سوف نقوم بإضافة (٥) مسافات بالمسطرة حتى يصبح عرضه (١٥) حرفا. ويتم كتابة الآتى فى العمود الثالث.

PHONE + SPACE (5) + CITY

وعند طباعة التقرير فإنه يأخذ الشكل التالى :

Hazem Kamel	Gama	2645321
	16 - Altayran	Cairo

ولإنشاء العناوين البريدية (Mailing Labels) يستخدم الأمر (CREATE) أو (MODIFY) أيضا ويتم تحديد إسم الملف وليكن (MAILLAB.LBL) ويمكن إدخال بيانات النموذج كالآتى :

Label contents 1 : name
2 : company
3 : address
4 : city

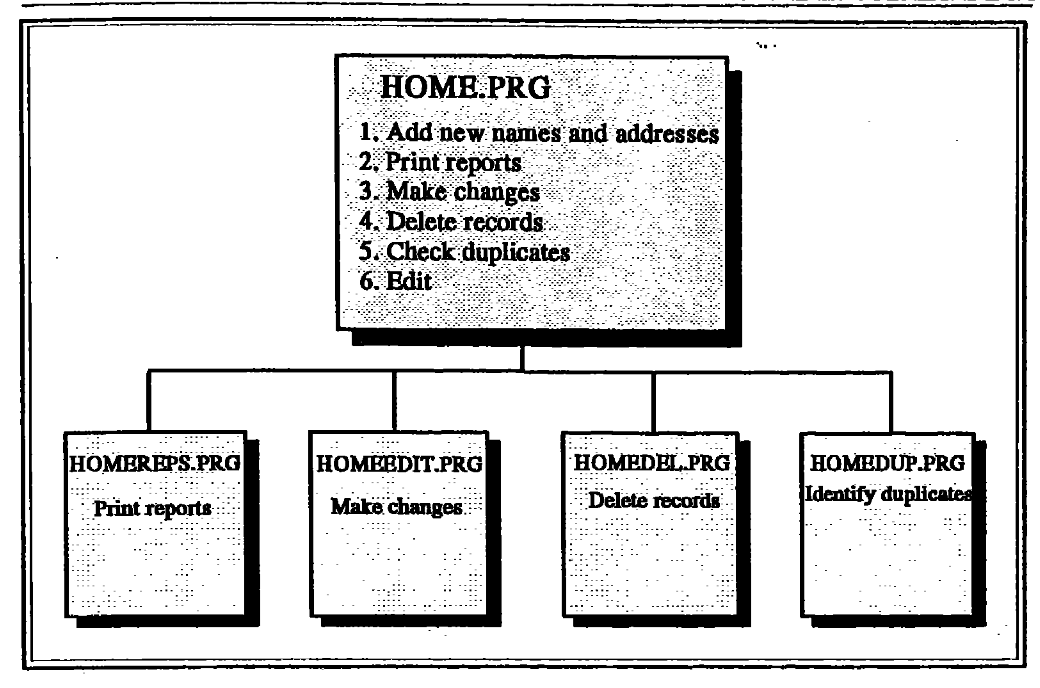
وعند عرض العناوين البريدية تظهر كالآتى :

Hazem Kamel
Gama
16 - Altayran
Cairo

Ayman Salah
Elnasr
16-Ahram street
Cairo

٢٧ - ٤ تركيب البرنامج

كما سبق أن أوضحنا في الأجزاء السابقة فإن تصميم نظام المعلومات يبدأ برسم التركيب الهرمى (Hierarchical Structure) للنظام وذلك كما يتضح من الشكل (٢٧ - ٧).



شكل (٢٧ - ٧)

ويلاحظ من الشكل أن النظام ينقسم إلى وظائف رئيسية (Tasks) يمثل كلا منها ملف أوامر (Command File). والبرنامج (.Home.prg) هو برنامج القائمة الرئيسية (Main Menu) ويكون فى قمة الهرم. والبرامج الأخرى تتفرع منه ويتم تشغيلها بواسطة الأمر (DO). وعند إنتهاء أى برنامج فرعى ينتقل التحكم إلى البرنامج الرئيسى. ويجب ملاحظة أن تقسيم النظام إلى وظائف منفصلة يؤدى إلى تبسيط عملية تخطيط البرامج. وذلك لأن تقسيم البرنامج إلى برامج صغيرة يعنى تصميم وتطوير واختبار كل برنامج مستقلا. وفى الأجزاء التالية يتم شرح برامج النظام بالتفصيل.

٢٧ - ٤ - ١ برنامج القائمة الرئيسية (HOME.PRG)

برنامج القائمة الرئيسية هو البرنامج الذى يقوم بتشغيل القائمة الرئيسية للنظام. وهو لايختلف فى تركيبه عن أى برنامج رئيسى يتم من خلاله عرض قائمة إختيارات. وكالعادة قبل كتابة أى برنامج يفضل كتابة الخطوات الأولية (Pseudocade) بأى لغة يجيدها مخطط البرامج حتى يمكن بعد ذلك كتابة الأوامر بلغة برامج عائلة (DBase) التى تنفذ هذه الخطوات. والخطوات الأولية لبرنامج القائمة الرئيسية (Home.prg) تتلخص فى الأتى :

١ - تجهيز بيئة البرنامج.
٢ - فتح ملف البيانات وملفات الفهرس.
٣ - تكوين حلقة تكرارية لعرض القائمة الرئيسية للبرنامج.
٤ - مسح الشاشة
٥ - عرض القائمة الرئيسية التى تكون كالآتى :

Home System Main Menu

1. Add new Names and Addresses
2. Print Directory or Labels
3. Make Changes
4. Delete Names and Addresses
5. Check for Duplicate Entries
6. Exit the Home System

٦ - إستقبال إختيار المستخدم
٧ - التفرع إلى البرنامج المطلوب
٨ - العودة إلى القائمة
٩ - الخروج من البرنامج فى حالة الإختيار رقم (٦)

وبعد كتابة الخطوات الأولية (Pseudocode) نبدأ فى كتابة البرنامج الذى يتكون من الأوامر التالية :

```
***** HOME.PRG
***** Home Management System : DBase III +
*- - - - - set up initial parameters
SET TAlk OFF
SET STATUS OFF
SET DEFAULT TO C
*- - - - Open the database and index files.
USE HOMEDATA INDEX  HOMENAME, HOMECITY
* - - - - - Begin loop for main menu.
```

```
CHOICE = 0
DO WHILE CHOICE < > 6
     CLEAR
     TEXT

                    Home System Main Menu

          1. Add New Names and Addresses
          2. Print Directory or Labels
          3. Make Changes
          4. Delete Names and Addresses
          5. Check for Duplicate Entries
          6. Exit the Home System

     ENDTEXT
     * - - - - - Get user's choice.
     @  16,20  SAY "Enter choice" GET CHOICE  ;
          PICTURE "9" RANGE  1,6
     READ

     * - - - - Branch to appropriate task or program.
     DO CASE
     CASE CHOICE = 1
          SET FORMAT TO HOMESCRN
          APPEND
          CLOSE FORMAT
     CASE CHOICE = 2
          DO HOMEREPS
     CASE CHOICE = 3
          DO HOMEEDIT
     CASE  CHOICE = 4
          DO HOMEDEL
```

```
        CASE CHOICE = 5
              DO HOMEDUP
        ENDCASE
ENDDO ( while choice < > 6 )
* - - - - Done with Program. Return to dot prompt.
SET  TALK ON
SET STATUS ON
RETURN
```

والبرنامج يبدأ كالعادة بكتابة إسم البرنامج ووظيفته ثم يتم تجهيز بيئة البرنامج عن طريق مجموعة من أوامر (SET) وهى كالآتى :

```
SET TALK OFF
SET  STATUS OFF
SET  DEFAULT TO  C
```

والجزء التالى من البرنامج يقوم بفتح ملف قاعدة البيانات (HOMEDATA.DBF) وملفات الفهرس (HOMENAM.NDX) و (HOMECITY.NDX) وذلك من خلال السطرين التاليين :

```
* - - - - Open the database and index files.
USE MOHEDATA INDEX  HOMENAME, HOMECITY
```

والجزء التالى من البرنامج يحتوى على الحلقة التكرارية (DO WHILE) التى تستمر فى عرض قائمة الإختيارات حتى يختار المستخدم الإختيار رقم (٦) فيتم توقف الحلقة. والأمران (TEXT) و (ENDTEXT) يؤديان إلى عرض السطور المحصورة بينهما على الشاشة. ويتكون هذا الجزء من السطور التالية :

```
* - - - - - Begin loop for main menu.
CHOICE = 0
DO WHILE CHOICE < > 6
     CLEAR
```

```
TEXT

                Home System Main Menu

            1. Add new Names and Addresses
            2. Print Directory or Labels
            3. Make Changes
            4. Delete Names and Addresses
            5. Check for Duplicate Entries
            6. Exit the Home System

ENDTEXT
```

والجزء التالى من البرنامج يتم من خلاله إستقبال إختيار المستخدم وتخزينه فى المتغير (CHOICE) وهو يفرض على المستخدم كتابة رقم محصور بين (١) ، (٦) وأى رقم خارج هذا المدى لايقبله البرنامج ويتيح للمستخدم المحاولة مرة ثانية. ويحتوى هذا الجزء على السطور التالية :

```
* - - - - Get user's choice
@ 16,20 SAY "Enter choice" Get CHOICE;
   PICTURE "9" RANGE 1,6
READ
```

والجزء التالى من البرنامج يحتوى على الحلقة (DO CASE) التى تحدد ما يجب عمله مع كل اختيار للمستخدم. فإذا اختار المستخدم الإختيار رقم (١) لإضافة سجلات جديدة فإن البرنامج يفتح شاشة الإدخال (HOMESCRN.FMT) ويتيح للمستخدم إضافة سجلات جديدة عن طريق الأمر (APPEND) وبعد الإنتهاء يتم إغلاق شاشة الإدخال عن طريق الأمر (CLOSE FORMAT). والإختيارات الأخرى تؤدى إلى التفرع إلى برامج خارجية عن طريق الأمر (DO). ويتكون هذا الجزء من السطور التالية :

```
* - - - - Branch to appropriate task or program.
```

```
DO CASE
    CASE CHOICE = 1
        SET FORMAT TO HOMESCRN
        APPEND
        CLOSE FORMAT
    CASE CHOICE = 2
        DO HOMEREPS
    CASE CHOICE = 3
        DO HOMEEDIT
    CASE  CHOICE = 4
        DO HOMEDEL
    CASE CHOICE = 5
        DO HOMEDUP
ENDCASE
```

والجزء الأخير من البرنامج يؤدى إلى إنهاء الحلقة التكرارية ومسح الشاشة والعودة إلى نظام التشغيل ويتكون من السطور التالية :

```
ENDDO ( while choice < > 6 )
* - - - - Done with Program. Return to dot prompt.
SET  TALK  ON
SET STATUS  ON
QUIT
```

٢٧ - ٤ - ٢ برنامج التقارير (HOMEREPS.PRG)

برنامج التقارير يؤدى إلى عرض قائمة إختيارات التقارير التى تتفرع إلى قوائم فرعية أخرى كما سيتم الإيضاح. وقبل كتابة البرنامج يتم كتابة الخطوات الأولية (Pseudocode) وهى كالآتى :

١ - يتم مسح الشاشة وعرض قائمة إختيارات التقارير التالية :

1. Directory
2. Mailing Labels
3. Return to menu

٢ - يتم الرجوع إلى القائمة الرئيسية فى حالة اختيار المستخدم الإختيار رقم (٣).

٣ - فى حالة إختيار المستخدم الإختيار (١) أو الإختيار رقم (٢) يتم عرض قائمة إختيارات نوع الترتيب (Sort Order) وتحديد النوع المطلوب. وتتكون هذه القائمة من الإختيارات التالية :

1. Alphabetical Order by Name
2. City Order
3. Original Order

٤ - يتم فتح ملفات الفهرس بناء على اختيار نوع الترتيب وذلك كالآتى :

أ - فى حالة الترتيب الهجائى (Alphabetical) يتم فتح الملف (HOMENAME.NDX).

ب - فى حالة الترتيب الأصلى (Oringinal) لا يتم فتح أى ملف فهرس.

٥ - يتم عرض اختيارات البحث (Query) ثم عرض نموذج البحث (Query Form) بناء على طلب المستخدم وترشيح قاعدة البيانات بناء على نموذج البحث.

٦ - يتم السؤال عن تجهيز الطابعة.

٧ - يتم طباعة التقرير حسب نوع المطلوب كالآتى :

أ - إذا كان المطلوب طباعة الدليل (Directory) يتم فتح ملف التشكيل (NEWDIREC).

ب - إذا كان المطلوب طباعة عناوين بريدية يتم فتح ملف التشكيل (MAILLAB).

٨ - يتم إيقاف الشاشة فى حالة عدم إختيار الطباعة.

٩ - يتم إغلاق ملفات الترشيح وملفات الفهرس.

١٠- يتم الرجوع إلى القائمة الرئيسية.

وبعد الإنتهاء من كتابة الخطوات الأولية يتم كتابة البرنامج الذى يتكون من السطور التالية :

```
******* HOMEREPS.PRG
* - - - - Reports Program for Home System

* - - - - Clear Screen and ask which report.
CLEAR
TEXT
                Select a Report Option

        1. Directory
        2. Mailing Labels
        3. Return to Main Menu
ENDTEXT
* - - -  Initialize variable and ask for report choice.
REPCHOICE = 0
@ 14 , 20 SAY "Enter your choice (1-3)" ;
        GET REPCHOICE PICTURE  "9" RANGE 1,3
READ
* - - - -  if return requested, return to Main Menu.
IF REPCHOICE  = 3
        RETURN
ENDIF
* - - - - Ask about sort order.
CLEAR
TEXT

                Select a Sort Order
        1. Alphabetical order by Name
        2. City  Order
        3. Original Order
```

```
ENDTEXT
* - - - - initialize variable and ask for sort choice.
SORTCHOICE = 0
@ 14 , 20  SAY " Enter your choice (1-3) " ;
        GET SORTCHOICE PICTURE "9" RANGE 1,3
READ
* - - - Use appropriate index file.
DO CASE
        CASE SORTCHOICE = 1
                SET INDEX TO HOMENAME
        CASE SORTCHOICE = 2
                SET INDEX  TO HOMECITY
        CASE SORTCHOICE = 3
                CLOSE INDEX
ENDCASE
* - - - - Ask about query.
CLEAR
QCHOICE = "A"
@ 10 , 5  SAY "Do you want (A) ll records , or a (Q)uery? " ;
        GET QCHOICE PICTURE "!"
READ
* - - - - Display query form if requested
IF QCHOICE = " Q "
        MODIFY QUERY HOMEGEN
        SET FILTER TO FILE HOMEGEN
ENDIF
* - - - - Ask about the printer.
PMACRO =  "   "
TOPRINT = " N "
CLEAR
@ 10 , 5  SAY " Send report to printer? (Y/N) " ;
        GET  TOPRINT PICTURE "!"
```

```
READ
* - - - -  Make a macro if Printer requested.
IF TOPRINT = " Y "
        PMACRO = " TO PRINT "
ENDIF
* - - - - Now print the report.
DO CASE
        CASE REPCHOICE = 1
                PEPORT FORM NEWDIREC & PMACRO
        CASE REPCHOICE = 2
                LABEL FORM MAILLAB & PMACRO
ENDCASE
*  - - - - if printer was not selected, pause
* - - - -  before returning to menu.
IF TOPRINT < > " Y "
        @  24 , 1 CLEAR
        WAIT " Press any key to return to menu. ."

ENDIF
* - - - - when report is done, set filter and
* - - - - index files back to normal , and then
* - - - - return to main menu.

SET FELTER TO
SET INDEX TO HOMENAME, HOMECITY
RETURN
```

ويبدأ البرنامج بكتابة إسم البرنامج ووظيفته ثم أمر مسح الشاشة وكتابة قائمة اختيارات التقارير بإستخدام الأمرين (TEXT) ، (ENDTEXT). ثم يتم سؤال المستخدم عن الإختيار المطلوب واستقبال إختيار المستخدم وتخزينه فى المتغير (REPCHOICE). ويتم ذلك من خلال السطور التالية :

```
*******  HOMEREPS. PRG
```

```
* - - - -  Reports program for Home System.
* - - - -  Clear screen and ask which report.
CLEAR
TEXT

            Select a Report Option
        1. Directory
        2.  Mailing Labels
        3.  Returen to Main Menu
ENDTEXT
* - - - -  initialize variable and ask for report choice.
REPCHOICE = 0
@  14 , 20  SAY " Enter your choice ( 1-3) " ;
        GET REPCHOICE PICTURE "9" RANGE 1,3
READ
```

والجزء التالى يعرض قائمة إختيارات الترتيب ثم الحلقة (DO CASE) التى يتم من خلالها فتح ملف الفهرس الخاص بكل نوع من الترتيب. ويتم ذلك من خلال السطور التالية :

```
* - - - - Ask about sort order.
CLEAR
TEXT

            Select a Sort Order
        1.  Alphabetical order by Name
        2.  City  Order
        3.  Original Order

ENDTEXT
* - - - - initialize variable and ask for sort choice.
SORTCHOICE = 0
```

```
@ 14 , 20 SAY " Enter your choice (1-3) " ;
      GET SORTCHOICE PICTURE " 9 " RANGE 1,3
READ
*---- Use appropriate index file.
DO CASE
      CASE SORTCHOICE = 1
            SET INDEX TO HOMENAME
      CASE SORTHOICE = 2
            SET INDEX TO HOMECITY
      CASE SORTCHOICE = 3
            CLOSE INDEX
ENDCASE
```

الجزء التالى من البرنامج يسأل اذا كان المستخدم يريد طباعة كل السجلات أو بعضها بناء على بحث (Query) معين. ويتم تخزين رد المستخدم فى المتغير (QCHOICE). ويستخدم التعبير ("!" PICTURE) فى تحويل رد المستخدم إلى حروف كبيرة (Upercase) بمجرد إدخالها. ويتم ذلك من خلال السطور التالية :

```
* ---- Ask about qurey.
ClEAR
QCHOICE = " A "
@ 10 , 5  SAy "Do you want (A) ll records, or a (Q) uery? " ;
GET QCHOICE PICTUTRE " ! "
READ
```

وإذا اختار المستخدم البحث (Query) يتم عرض نموذج البحث (HOMEGEN.QRY) على الشاشة حتى يقوم المستخدم بملىء النموذج. ثم يتم ترشيح قاعدة البيانات بإستخدام الأمر (SET FILTER TO FILE) ويتم ذلك من خلال السطور التالية :

```
* ---- Display query form if requested.
IF QCHOICE = " Q "
```

```
        MODIFY QUERY HOMEGEN
        SET FILTER TO FILE HOMEGEN
ENDIF
```

والجزء التالى يسأل المستخدم اذا كان يريد طباعة التقرير أم الإكتفاء بعرضه على الشاشة. ويتم تخزين رد المستخدم فى المتغير (TOPRINT). والسطور التالية توضح هذا الجزء.

```
* - - - - Ask about the printer.
PMACRO = "  "
TOPRINT = " N "
CLEAR
@ 10 , 5  SAY "Send report to Printer? (Y/N) " ;
        GET TOPRINT PICTURE  " !"
READ
```

والجزء التالى من البرنامج يؤدى إلى طباعة التقرير المطلوب بناء على إختيار المستخدم المخزن فى المتغير (REPCHOICE). والحلقة (DO CASE) تحدد نوع التقرير المطبوع. ويجب ملاحظة إستخدام الماكرو (&PMACRO). فإذا طلب المستخدم طباعة التقرير فإن العبارة (TOPRINT) تضاف إلى الأمر (REPORT) أو الأمر (LABEL) مما يؤدى إلى الحصول على المخرجات المطلوبة. وإذا لم يطلب المستخدم طباعة التقرير على الطابعة يصبح الماكرو خاليا وبالتالى لايتم كتابة أى شىء بعد الأمر (REPORT) أو الأمر (LABEL) وبالتالى تظهر المخرجات على الشاشة فقط. ويتم ذلك من خلال السطور التالية :

```
* - - - - Now Print the report
DO CASE
        CASE REPCHOICE  = 1
            REPORT FORM NEWDIREC  & PMACRO
        CASE REPCHOICE  = 2
            LABEL FORM MAILLAB & PMACRO
ENDCASE
```

والجزء التالى يؤدى إلى توقف الشاشة لحظيا (Pause) قبل الرجوع إلى القائمة الرئيسية. وذلك فى حالة عدم رغبة المستخدم طباعة التقرير. وهذا يتيح له قراءة التقرير على الشاشة. والسطور التالية توضح هذا الجزء من البرنامج.

```
* - - - -  if printer was not selected, pause
* - - - -  before returning to menu.
IF TOPRINT < > " Y "
        @ 24 ,1  CLEAR
        WAIT " Press any key to return to menu ... "
ENDIF
```

والجزء التالى يؤدى إلى الرجوع إلى القائمة الرئيسية حيث يتم إغلاق المرشح (Filter) كما يتم فتح الفهارس مرة ثانية. والسطور التالية توضح هذا الجزء.

```
* - - - -  When report is done, set filter and
* - - - -  index files back to mormal, and then
* - - - -  return to main menu.
SET FILTER TO
SET INDEX TO  HOMENAME, HOMECITY
RETURN
```

٢٧ - ٤ - ٣ تصحيح البيانات

يتيح البرنامج (HOMEEDIT.PRG) للمستخدم إستدعاء أى سجل وتصحيح بياناته. والخطوات الأولية (Pseudocode) للبرنامج كالآتى :

١ - يتم تكوين حلقة تكرارية لتصحيح السجلات.
٢ - يتم السؤال عن الإسم المطلوب تصحيح بياناته ويكفى فى هذة الحالة كتابة أول حرف أو الحروف الأولى من الإسم.
٣ - إذا لم يتم إدخال أى إسم يتم الرجوع إلى القائمة الرئيسية.
٤ - يتم تحويل حروف المتغير (LOOKUP) إلى حروف كبيرة ليطابق ملف الفهرس.

٥ - يتم البحث عن الإسم المطلوب.
٦ - يتم حساب عدد السجلات التى تحتوى على الإسم المطلوب.
٧ - إذا لم تكن هناك سجلات تحتوى على هذا الإسم يطلب البرنامج من المستخدم إعادة المحاولة.
٨ - إذا كانت هناك عدة سجلات تحتوى على الإسم المطلوب يتم عرض بيانات هذا السجلات وسؤال المستخدم عن رقم السجل المطلوب.
٩ - عند تحديد السجل المطلوب يتم عرض بياناته على الشاشة حتى يستطيع المستخدم تعديل بياناته.
١٠- يتم العودة إلى القائمة الرئيسية بعد إدخال التعديلات المطلوبة.

والبرنامج الذى يحقق هذه الخطوات الأولية يتم كتابته كالآتى :

```
****** HOMEEDIT.PRG
* Look up and Edit Names on the HOMEDATA Database.

* - - - - Set up loop for editing records.
STILLATIT = .T .
DO WHILE  STILLATIT
        * - - - - Ask for name of person to look up.
        CLEAR
        LOOK UP = SPACE (15)
        @ 10 , 12 SAY " Enter name of person to edit "
        @ 12 ,12 SAY " or just press Return to exit " ;
            GET LOOKUP
        READ
        * - - - - IF no name entered skip all commands
        * - - - - between here and ENDDO.
        IF LOOKUP =  "   "
                STILLATIT = .F.
                LOOP
        ENDIF (LOOKUP = "  ")
```

```
* - - - - - Convert  Lookup to uppercase
* - - - - - to match index file.
LOOKUP = UPPER (LOOKUP)
* - - - - - Try to find requested name, and
* - - - - -  remember record number.
SEEK LOOKUP
RECNUMB = RECNO ( )
* - - - - Count how many there are.
COUNT WHILE UPPER ( NAME ) = LOOKUP TO  ;
HOWMANY
* - - - - IF no record has that name,
* - - - - ask the user to try again.
IF HOWMANY = 0
        @  20 , 10 SAY " There is no & Lookup "
        @  22 , 10 SAY " Press a key to try again"
        ? CHR (7)
        WAIT "  "
        RECNUMB = 0
ENDIF ( HOWMANY  =  0 ) .
* - - - - If more than one record have.
* - - - - that name, get more information.
IF HOWMANY > 1
        CLEAR
        RECNUMB = 0
        SEEK LOOKUP
        LIST NAME , ADDRESS , CITY ;
            WHILE UPPER (NAME) = LOOKUP
        @ REW (  ) + 3 , 10  SAY "Edit which record # ? " ;
            GET RECNUMB PICTURE "9999"
        READ
ENDIF(HOWMANY > 1)
* - - - if there is a record number greater than
```

```
        * - - - zero , edit the record
        IF RECNUMB > 0
                GO TO RECNUMB
                SET FORMAT TO HOMESCRN
                READ
                CLOSE  FORMAT
        ENDIF
ENDDO (while STILLATIT)
RETURN
```

وهذا البرنامج يوضح عددا من وسائل البرمجة الأساسية للوصول إلى البيانات باستخدم ملفات الفهرس. ويبدأ البرنامج بتوصيف البرنامج كالعادة. وتستخدم الحلقة التكرارية (DO WHILE) فى التحكم فى متغير منطقى يسمى (STILLATIT) كما يتضح من السطور التالية :

```
***** HOMEEDIT.PRG
***** Lookup and Edit Names on the HOMEDATA  database
* - - - -  Set up loop for editing records.
STILLATIT = .T.
DO WHILE STILLATIT
```

ومن خلال الحلقة التكرارية يتم إنشاء المتغير الحرفى (LOOKUP) الذى يكون طوله (١٥) حرفا خاليا (Spaces). ويتم سؤال المستخدم عن الإسم المطلوب ثم تخزين هذا الإسم فى المتغير (LOOKUP) كما يتضح من السطور التالية :

```
* - - - Ask for name of person to lookup.
CLEAR
LOOKUP = SPACE  (15)
@ 10 , 12  SAY " Enter name of person to edit "
@ 12 , 12  SAY " or just press Return to exit "  ;
   GET LOOKUP
READ
```

وإذا ضغط المستخدم على مفتاح الإدخال دون أن يكتب أى إسم فإن البرنامج يعرف أن المستخدم قد إنتهى من التعديل أو أنه لايريد التعديل وفى هذه الحالة يتم الرجوع إلى القائمة الرئيسية. ويتم ذلك عن طريق السيطرة على قيمة المتغير (STILLATIT) الذى يتحكم فى الحلقة التكرارية وإستخدام الأمر (LOOP) فى الرجوع إلى أولها دون تنفيذ باقى أوامر الحلقة. والسطور التالية توضح ذلك :

```
* - - - - if no name entered, skip all commands
* - - - - between here and ENDDO
IF LOOKUP = "   "
      STILLATIT = . F .
      LOOP
ENDIF ( LOOKUP =  " " )
```

وإذا أدخل المستخدم إسما فإن البرنامج يبحث عن هذا الإسم. ومما سبق يلاحظ أننا استخدمنا الدالة (UPPER) لتحويل كل الأسماء فى الفهرس (HOMENAME.NDX) إلى حروف كبيرة. لذلك يتم هنا إستخدام الدالة (UPPER) فى تحويل ما يدخله المستخدم إلى حروف كبيرة. وهذا يضمن أن مايدخله المستخدم يكون دائما حروفا كبيرة مطابقة للأسماء الموجودة فى الفهرس بصرف النظر عن طريقة كتابة المستخدم لها. فمثلا الإسم (Ahmed) يمكن أن يكتب (AHMED) أو (ahmed) أو (Ahmed) دون أن يؤثر ذلك على بحث البرنامج عن السجل الخاص بهذا الإسم. والسطور التالية توضح عملية تحويل المتغير (Lookup) إلى حروف كبيرة.

```
* - - - - Convert LOOKUP to upercase to match
* - - - - index file.
LOOKUP = UPPER (LOOKUP)
```

ثم يبدأ البرنامج البحث عن الإسم المطلوب فى ملف الفهرس (HOMENAME.NDX). ويقوم البرنامج بتخزين رقم السجل الذى يحتوى على هذا الإسم فى المتغير (RECNNMB) وذلك من خلال السطور التالية :

```
* - - - - - Try to find requested name, and
```

```
* - - - - - remember record number.
SEEK LOOKUP
RECNUMB = REC NO (   )
```

ولأن هناك إحتمال وجود أكثر من سجل له نفس الإسم لذلك يستخدم البرنامج الأمر (COUNT) فى حصر عدد السجلات المحتوية على الإسم المطلوب. ولأن قاعدة البيانات تكون مرتبة هجائيا (Alphabetically) حسب الإسم لذلك يفضل استخدام التعبير (WHILE) بدلا من التعبير (FOR) لأنه يكون أسرع لوجود السجلات المشتركة فى الإسم فى مكان واحد على القرص. ويقوم البرنامج بعد ذلك بتخزين عدد السجلات المحتوية على الإسم المطلوب فى المتغير (HOWMANY). والسطور التالية توضح ذلك.

```
* - - - - Count how many threre are.
COUNT WHILE UPPER ( NAME ) = LOOKUP TO HOWMANY
```

والجزء التالى من البرنامج يوضح إتخاذ القرار بناء على نتيجة البحث عن الإسم. فإذا كان الإسم المطلوب غير موجود فى قاعدة البيانات يقوم البرنامج بعرض رسالة خطأ (Error Message) مع إطلاق صفارة التحذير من خلال السطر ((7) CHR ?) ويتوقف البرنامج لحظيا فى انتظار ضغط المستخدم على أى مفتاح. والماكرو(&LOOKUP) يستخدم فى عرض الإسم الذى يتم إدخاله مع رسالة الخطأ. والسطور التالية توضح هذه الحالة.

```
* - - - - If no record has that name,
* - - - - ask the user to try again.

IF HOWMANY = 0
        @ 20 , 10  SAY " There is no  &LOOKUP "
        @ 22 , 10 " Press a key to try again "
        ? CHR (7)
        WAIT "  "
RECNUMB  = 0
ENDIF (HOWMANY = 0 )
```

وإذا كانت هناك عدة سجلات محتوية على الإسم المطلوب فإن البرنامج يريد معلومات أكثر لاختيار السجل المطلوب من بين هذه السجلات. لذلك يقوم البرنامج أولا بمسح الشاشة ثم يبحث عن أول سجل يحتوى على الإسم المطلوب (يلاحظ هنا تكرار عملية البحث وذلك لأن مؤشر السجل يكون قد انتقل بعد استخدام الأمر (COUNT) وأصبح فى نهاية الملف). ويستخدم الأمر (LIST) فى عرض بيانات السجلات المشتركة فى الإسم متضمنة رقم السجل (Record Number) كما يتم عرض رسالة على الشاشة تطلب من المستخدم كتابة رقم السجل المطلوب. والسطور التالية توضح هذا الجزء من البرنامج.

```
* - - - - If more than one record has that
* - - - - name, get more information.
IF HOWMANY > 1
        CLEAR
        RECUNMB = 0
        SEEK LOOKUP
        LIST NAME , ADDRESS, CITY ;
                WHILE UPPER ( NAME ) = LOOKUP
        @ ROW ( ) + 3,10  SAY " Edit which record # ? " ;
                GET RECUNMB PICTURE " 9999 "
        READ
ENDIF
```

ويلاحظ أنه أصبحت لدينا حالة من إثنين ، إما أن هناك سجلا أصبح جاهزا للتعديل ورقمه مخزن فى المتغير (RECNUMB) أو أنه ليست هناك أى سجلات محتوية على الإسم المطلوب وبالتالى تكون قيمة المتغير (RECNUMB) مساوية للصفر (Zero). واذا كان هناك سجل جاهز للتعديل يتم فتح ملف التشكيل (HOMESCRN.FMT) واستقبال تعديلات المستخدم على السجل. والسطور التالية توضح هذا الجزء.

```
* - - - - If threre is a record number
* - - - - greater than zero, edit the record
IF RECNUMB > 0
```

```
GOTO  RECNUMB
      SET FORMAT TO HOMESCRN
      READ
      CLOSE FORMAT
ENDIF
```

والجزء الأخير من البرنامج يوضح أوامر إغلاق الحلقة التكرارية والعودة إلى القائمة الرئيسية فى حالة توقف المستخدم عن طلب تعديل سجلات أخرى. والسطور التالية توضح ذلك.

```
ENDDO ( while STILLATIT )
RETURN
```

٢٧ - ٤ - ٤ مسح السجلات

برنامج مسح السجلات (HOMEDEL.PRG) يشبه إلى حد كبير برنامج التعديل (HOMEEDIT.PRG) وذلك من حيث استدعاء سجل معين يطلبه المستخدم لتنفيذ عملية معينة عليه. والخطوات الأولية (Pseudacode) تتلخص فى الآتى :

١ - يتم تكوين حلقة تكرارية لمسح السجلات.
٢ - يتم السؤال عن إسم الشخص المطلوب مسح سجله.
٣ - إذا ضغط المستخدم على مفتاح الإدخال يتم الرجوع إلى القائمة الرئيسية.
٤ - يتم تحويل المتغير (LOOKUP) إلى حروف كبيرة لتطابق ملف الفهرس.
٥ - يتم البحث عن الإسم المطلوب.
٦ - يتم حصر عدد السجلات المحتوية على الإسم المطلوب.
٧ - إذا لم تكن هناك أى سجلات محتوية على الإسم المطلوب يتم تحذير المستخدم والسماح له بالمحاولة مرة أخرى.
٨ - إذا كانت هناك عدة سجلات محتوية على الإسم المطلوب يطلب البرنامج من المستخدم معلومات أكثر عن السجل المطلوب.
٩ - إذا تم تحديد السجل المطلوب مسحه يطلب البرنامج من المستخدم الإذن له بمسح هذا السجل.

١٠ - إذا وافق المستخدم على المسح يتم تمييز هذا السجل لمسحه فيما بعد بإستخدام الأمر (PACK).
١١ - يسمح البرنامج للمستخدم بتكرار عملية المسح لسجلات أخرى حتى يطلب الخروج من البرنامج.
١٢ - يتم حصر عدد السجلات التى تم تمييزها للمسح.
١٣ - طالما كانت هناك سجلات تم تمييزها للمسح يتم تنفيذ الآتى :
أ - يتم عرض السجلات المميزة للمسح.
ب - يطلب البرنامج من المستخدم الإذن بمسح هذه السجلات نهائيا.
حـ - اذا لم يوافق المستخدم على مسح كل هذه السجلات يتم تنفيذ الآتى:
١ - يسمح البرنامج للمستخدم باستعادة أحد السجلات التى لايريد مسحها.
٢ - يتم إنقاص عداد السجلات المميزة للمسح بواحد.
د - اذا وافق المستخدم على مسح السجلات المميزة للمسح يتم مسح السجلات نهائيا باستخدام الأمر (PACK).
هـ - عند الإنتهاء من مسح السجلات يتم الرجوع إلى القائمة الرئيسية.

والبرنامج الذى يحقق هذه الخطوات الأولية يتم كتابته كالأتى :

```
****** HOMEDEL.PRG
* Lookup and Delete  Names in the HOMEDATA database.

* - - - - - Setup loop for deleting records.
STILLATIT = .T.
DO WHILE STILLATIT
        * - - - - Ask for name of person to lookup.
        CLEAR
        LOOKUP = SPACE (15)
        @  10 ,12  SAY " Enter name of person to delete "
        @  12 ,12  SAY " or just press Return to exit ";
              GET LOOKUP
        READ
        * - - - - If no name entered , skip all commands
```

```
* - - - - between here and ENDDO.
IF LOOKUP = "    "
        STILLATIT = .F.
        LOOP
ENDIF (LOOKUP = "    ")
* - - - - - Convert LOOKUP to uppercase to match
* - - - - - index file.
LOOKUP = UPPER (LOOKUP)
* - - - - try to find the requested name , and
* - - - - remember record number
SEEK LOOKUP
RECNUMB = RECNO ( )
* - - - - If no record has that name , warn
* - - - - the user to try again.
* - - - count how many there are
COUNT WHILE UPPER (NAME) = LOOKUP TO  ;
HOWMANY
IF HOWMANY = 0
        @ 20 , 10  SAY  " There is no & Lookup "
        @ 22 , 10  SAY " Press a key to try again "
        ? CHR (7)
        WAIT "  "
        RECNUMB = 0
ENDIF ( HOWMANY = 0 )
* - - - - If more than one record has that
* - - - - name , get more information.
IF HOWMANY > 1
        CLEAR
        RECNUMB = 0
        SEEK LOOKUP
        LIST NAME , ADDRESS , CITY ;
                WHILE UPPER ( NAME) = LOOKUP
```

```
        @ ROW ( ) + 3,10 SAY " Delete which one ? " ;
              GET RECNUMB PICTURE " 9999"
        READ
    ENDIF(HOWMANY > 1)
    * - - - - If there is a record number greater than
    * - - - - zero , then double check and delete.
    IF RECNUMB > 0
        GOTORECNUMB
        CLEAR
        DISPLAY NAME , ADDRESS , CITY
        ?
        WAIT " Delete this record ? (Y/ N) " TO ANSWER
        * - - - - If answer is yes , mark record for deletion.
        IF UPPER ( ANSWER) = " Y "
            DELETE RECORD RECNUMB
        ENDIF ( ANSWER)
    ENDIF (RECNUMB > 0)
ENDDO ( while STILLATIT)

* - - - - Before exiting , verify deletions and pack.
COUNT FOR DELETED ( ) TO NODELS
OKTOPACK = " N "
DO WHILE OKTOPACK = " N " . AND. NODELS > 0
    CLEAR
    ? " Records to be deleted ... '
    ?
    DISPLAY NAME , ADDRESS FOR DELETED ( )
    @ 23,1 SAY " Delete all these ? (Y / N) " ;
          GET OKTOPACK PICTURE " !"
    READ
    IF OKTOPACK < > " Y "
        *- - - - If not ok to pack , recall a record.
```

```
            DELREC = 0
            @23,1 SAY "Recall which one(byrecord #)" ;
                  GET DELREC PICTURE " 9999 "
            READ
            * - - - - If record number entered , and record
            * - - - - is indeed deleted, recall it.
            IF DELREC > 0
                  GOTO  DELREC
                  IF ELETED ( )
                        RECALL RECORD  DELREC
                        NODELS = NODELS - 1
                  ENDIF ( DELETED )
            ENDIF ( DELREC > 0 )
      ELSE
            *- - If ok to pack , do so and show progress.
            SET TALK ON
            PACK
            SET TALK OFF
      ENDIF ( OKTOPACK)
ENDDO (OKTOPACK)
RETURN
```

والتكنيك العام للبحث عن سجل معين لمسحه لايختلف عن التكنيك الذى سبق شرحه فى برنامج (HOMEEDIT.PRG). ولكن عند الوصول إلى السجل المطلوب فإن هذا البرنامج لايسمح بتعديله ولكنه يعرض جزءا من بيانات هذا السجل ويطلب من المستخدم الإذن بالمسح. وإذا كتب المستخدم (Y) فإن الأمر (DELETE RECORD) يقوم بتمييز هذا السجل لمسحه فيما بعد بواسطة الأمر (PACK). والسطور التالية توضح هذا الجزء :

```
* - - - -  If there is a record number greater
* - - - - than  zero , then double check and delete.
IF RECNUMB > 0
```

```
        GOTO  RECNUMB
        CLEAR
        DISPLAY NAME , ADDRESS , CITY
        ?
        WAIT " Delete this record ? (Y / N) " TO ANSWER
        * - - - - If answer is yes ,
        * - - - - mark record for deletion.
        IF UPPER ( ANSWER) = " Y "
              DELETE RECORD RECNUMB
        ENDIF ( ANSWER)
ENDIF ( RECNUMB > 0 )
```

والبرنامج يتيح للمستخدم مسح أى عدد من السجلات. وقبل الرجوع إلى القائمة الرئيسية يقوم بالتأكد من رغبة المستخدم مسح هذه السجلات كما يتيح له إستعادة أى سجل يزيد التراجع عن مسحه وذلك باستخدام الأمر (Count). كما يقوم بتخزين هذا العدد فى المتغير (NODELS).

ثم تقوم الحلقة التكرارية (DO WHILE) بعرض السجلات التى تم تمييزها للمسح وتسأل المستخدم (Delete all these ?) فإذا كانت إجابته (NO) فإن البرنامج يتيح للمستخدم إستعادة أى سجل من هذا السجلات وذلك بواسطة رقم هذا السجل. ثم يقوم البرنامج بعرض باقى السجلات التى تم تمييزها للمسح ويعيد نفس الخطوات السابقة. وتستمر هذه العملية حتى يحدث شىء من إثنين فإما أن يجيب المستخدم بنعم (Yes) على السؤال (Delete all these?) أو لايبقى هناك سجلات جاهزة للمسح (NODELS = 0). وفى هذه الحالة يستخدم الأمر (PACK) فى مسح السجلات نهائيا. ولأن عملية المسح النهائى للسجلات تأخذ وقتا طويلا نسبيا (وذلك لأن سجلات قاعدة البيانات يعاد نسخها ما عدا السجلات الممسوحة) لذلك يتم استخدام الأمر (SET TALK ON) فى عرض الرسائل الدالة على اجراء عملية المسح حتى لايشك المستخدم فى توقف البرنامج. وعند إنتهاء عملية المسح (Packing) يتم إستخدام الأمر (SET TALK OFF) لإعادة البرنامج إلى حالته السابقة. والسطور التالية توضح هذا الجزء من البرنامج.

```
* - - - - Before exiting , verify deletions and pack.
```

```
COUNT FOR DELETED ( ) TO NODELS
OKTOPACK = " N "
DO WHILE OKTOPACK = " N " .AND. NODELS > 0
      CLEAR
      ? " Records to be deleted ... '
      ?
      DISPLAY NAME , ADDRESS FOP DELETED (  )
      @ 23,1  SAY " Delete all these ? (Y / N) " ;
            GET OKTOPACK PICTURE " !"
      READ
      IF OKTOPACK < >  " Y "
            *- - - - If not ok to pack , recall a record.
            DELREC = 0
            @ 23,1  SAY " Recall which one ( by record #) ;
                  GET DELREC PICTURE " 999 "
            READ
            * - - - - If record number entered , and record
            * - - - - is indeed deleted, recall it.
            IF DELREC > 0
                  GOTO DELREC
                  IF DELETED ( )
                        RECALL RECORD  DELREC
                        NODELS = NODELS - 1
                  ENDIF ( DELETED )
            ENDIF ( DELREC > 0 )
      ELSE
            * - - If ok to pack , do so and show progress.
            SET TALK ON
            PACK
            SET TALK OFF
      ENDIF ( OKTOPACK)
ENDDO ( OKTOPACK)
RETURN
```

٢٧ - ٤ - ٥ إختبار التكرار

برنامج إختبار التكرار (HOMEDUP.PRG) يؤدى إلى عرض السجلات التى تحتوى على حقول متماثلة فيما بينها مثل حقول الإسم والعنوان والمدينة وذلك حتى يتيح للمستخدم مسح السجلات المكررة والخطوات الأولية للبرنامج تتلخص فى الآتى :

١ - السؤال عن تجهيز الطابعة.
٢ - يتم ترتيب قاعدة البيانات حسب المدينة والعنوان والإسم وتخزينها فى ملف مؤقت (Temporary).
٣ - فى حالة طلب الطباعة يتم تشغيل الطابعة
٤ - يتم طباعة عنوان التقرير.
٥ - يتم تكوين حلقة تكرارية لفحص سجلات قاعدة البيانات.
٦ - يتم مقارنة حقول المدينة والعنوان والإسم فى كل سجل والسجل الذى يليه.
٧ - فى حالة تطابق سجلين متتالين يتم الرجوع إلى أول سجل مطابق.
٨ - يتم عرض جميع السجلات المتطابقة.
٩ - عند الإنتهاء من عرض السجلات يتم توجيهها إلى الشاشة أو إلى الطابعة.
١٠ - يتم مسح ملف الفهرس المؤقت وإعادة فهرسة قاعدة البيانات كما كانت.
١١ - يتم الرجوع إلى القائمة الرئيسية.

والبرنامج الذى يحقق هذه الخطوات الأولية يتم كتابة كالآتى :

```
******** HOMEDUP.PRG
*- - - - - Home System check for duplication program.
* - - - -- Ask about printer.
CLEAR
TOPRINT = " N "
@ 10 , 10  SAY " Send duplicates to printers? (Y/ N) " ;
      GET TOPRINT PICTURE " ! "
READ

* - - - - - -Display Opening Messages and show progress
@ 15 , 10  SAY " Sorting for duplicates check ... "
SET TALK ON
```

```
USE HOMEDATA
INDEX ON UPPER ( CITY + ADDRESS + NAME ) TO  TEMP
SET TALK OFF
CLEAR
* - - - - IF printer requested turn it on.
IF TOPRINT = " Y "
        SET PRINT ON
ENDIF
* - - - - Print report title.
CLEAR
? DTOC ( DATE ( )) + SPACE ( 20 )+ " Possible Duplications "
? ?  SPACE ( 20) + TIME ( )
?
?
* - - - - - Loop through the database.
GO TOP
DO WHILE . NOT . EOF (   )
        COMPARE  = UPPER ( CITY + ADDRESS + NAME )
        SKIP
        IF UPPER ( CITY + ADDRESS + NAME) = COMPARE
                SKIP - 1
                LIST NAME, ADDRESS , CITY WHILE ;
                 UPPER ( CITY + ADDRESS + NAME ) = COMPARE
                ?
                ?
        ENDIF ( UPPERCITY + - - - )
ENDDO ( not eof )

* - - - Done with report, handle printer or screen.
IF TOPRINT = " Y "
        EJECT
        SET PRINT OFF
```

```
ELSE
	@ 23, 1
	WAIT "Press a key to return to the menu ... "
ENDIF (TOPRINT)
* - - - - Erase temporary index file and
* - - - - reactivate normal indexes.
CLOSE DATABASES
ERASE TEMP. NDX
USE HOMEDATA INDEX HOMENAME , HOMECITY
```

والجزء الأول من البرنامج يعيد ترتيب قاعدة البيانات بناء على المدينة العنوان والإسم وبذلك تصبح السجلات المتماثلة في هذه الحقول الثلاثة مرتبة وراء بعضها. ويتم ترتيب قاعدة البيانات باستخدام الأمر (INDEX) عن طريق الملف الفهرسى (TEMP.NDX). ويلاحظ في بداية هذا الجزء أن البرنامج يسأل إذا كان المطلوب طباعة هذه السجلات أم الإكتفاء بعرضها على الشاشة. ويتم من خلال السطور التالية :

```
******** HOMEDUP.PRG
*- - - - - Home System check for duplication program.
* - - - Ask about printer.
CLEAR
TOPRINT = " N "
@ 10 , 10 SAY " Send duplicates to printers? (Y / N) " ;
	GET TOPRINT PICTURE  " ! "
READ
* - - - - - -Display opening Messages and show progress
@ 15 , 10 SAY " Sorting for duplicates check ... "
USE HOMEDATA
INDEX ON UPPER (CITY + ADDRESS + NAME ) TO TEMP
SET TALK OFF
CLEAR
```

وإذا طلب المستخدم طباعة التقرير فإن الجزء التالى من البرنامج يؤدى إلى تشغيل الطابعة. ويجب ملاحظة أن الأمر (SET PRINT ON) يؤدى إلى إرسال أي مخرجات إلى الطابعة ماعدا مخرجات الأمر (@) التى يلزم لإرسالها إلى الطابعة إستخدام الأمر (SET DEVICE TO PRINT) والسطور التالية توضح هذا الجزء من البرنامج.

```
* - - - - IF printer requested turn it on .
IF TOPRINT = " Y "
      SET PRINT ON
ENDIF
```

والجزء التالي من البرنامج يؤدى إلى طباعة تاريخ اليوم الحالى والوقت الحالى ثم عنوان التقرير وسطرين خاليين. وذلك من خلال السطور التالية :

```
* - - - - Print report title.
CLEAR
? DTOC ( DATE ( )) + SPACE (20 ) + " Possible Duplications "
?? SPACE ( 20 ) + TIME ( )
?
?
```

والجزء التالى يتم من خلاله تكوين حلقة تكرارية لقراءة سجلات قاعدة البيانات بدءا من أول سجل. وذلك من خلال السطور التالية.

```
* - - - - Loop through the database.
GO TOP
DO WHILE . NOT. EOF ( )
```

والجزء التالى يتم من خلاله إنشاء المتغير الحرفى (COMPARE) الذى يحتوى على مجموع حقول المدينة والعنوان والإسم بعد تحويلها إلى حروف كبيرة. ويتم ذلك من خلال السطر التالى :

```
COMPARE = UPPER ( CITY + ADDRESS + NAME )
```

والجزء التالى يتم من خلاله إستخدام الأمر (SKIP) فى تحريك المؤشر إلى السجل التالى ويستخدم الأمر (IF) فى تحديد ما اذا كانت حقول المدينة والعنوان والإسم للسجل الحالى مطابقة لنفس الحقول الخاصة بالسجل السابق أم لا. ونظرا لأن قاعدة البيانات تكون مرتبة تبعا لهذه الحقول فإن السجلات المطابقة تكون وراء بعضها فى نفس المكان. فإذا تطابق سجلان فإن البرنامج يعيد المؤشر إلى السجل المطابق الأول ويبدأ فى عرض جميع السجلات المطابقة لهذا السجل. والسطور التالية توضح هذا الجزء من البرنامج.

```
SKIP
        IF UPPER ( CITY + ADDRESS + NAME ) = COMPARE
        SKIP - 1
        LIST NAME, ADDRESS , CITY WHILE ;
           UPPER ( CITY + ADDRESS + NAME ) = COMPARE
        ?
        ?
ENDIF (UPPER CITY + - - - - )
```

والجزء التالى من البرنامج يتم من خلاله إستمرار تنفيذ الحلقة التكرارية حتى نهاية الملف. واذا تم طباعة التقرير يتم نقل الورقة بواسطة الأمر (EJECT) ثم يتم إغلاق الطابعة. واذا تم عرض التقرير على الشاشة فقط فإن البرنامج يتوقف لحظيا حتى يسمح للمستخدم بقراءة الشاشة قبل العودة إلى القائمة الرئيسية. والسطور التالية توضح هذا الجزء من البرنامج.

```
ENDDO ( Not eof )
* - - - Done with report, handle printer or screen.
IF  TOPRINT = " Y "
       EJECT
       SET PRINT OFF
ELSE
       @ 23, 1
       WAIT "Press a key to return to the menu ... "
ENDIF (TOPRINT)
```

والجزء الأخير من البرنامج يتم من خلاله إغلاق ملفات قواعد البيانات والفهارس المفتوحة كما يتم مسح الملف المؤقت (TEMP.NDX) من القرص وإعادة الفهارس إلى حالتها الأولى قبل تنفيذ هذا البرنامج.

```
* - - - - Erase temporary index file and
* - - - - reactivate normal indexes.
ClOSE DATABASES
ERASE TEMP. NDX
USE HOMEDATA INDEX HOMENAME , HOMECITY
RETURN
```

الفصل الثامن والعشرون

مــولــد التطبيقــات

(Application Generator)

مولد التطبيقات الخاص ببرنامج (+ DBase III) هو برنامج يصمم البرامج للمستخدم ويغنيه عن كتابة أى كود. وفى هذا الفصل يتم تصميم نظام لتسجيل الشيكات من خلال مولد التطبيقات.

٢٨ - ١ تشغيل مولد التطبيقات

لتشغيل مولد التطبيقات يتم كتابة الأمر التالى :

DO APPSGEN

وهذا يؤدى إلى ظهور قائمة تحتوى على عشرة إختيارات كالآتى :

1. CREATE DATABASE
2. CREATE SCREEN FORM
3. CREATE REPORT FORM
4. CREAT LABEL FORM
5. SET APPLICATION COLOR
6. AUTOMATIC APPLICATION GENERATOR
7. RUN APPILCATION
8. ADVANCED APPLICATION GENERATOR
9. MODIFY APPLICATION CODE
10. EXIT

ويختار المستخدم أى اختيار عن طريق كتابة رقم هذا الإختيار والضغط على مفتاح الإدخال. ويلاحظ أن الاختيارات من (1) إلى (5) هى عمليات معروفة سبق شرحها. فمثلا عندما يختار المستخدم الرقم (١) يظهر السؤال التالى على الشاشة.

Enter the name of the new file.

ويمكن فى هذه الحالة كتابة الإسم (CHECKS) ثم ادخال تركيب قاعدة البيانات كالموضح بالشكل (٢٨ - ١) كما يتم تخزين قاعدة البيانات.

```
CURSOR  <--  -->    INSERT         DELETE        Up a field:    ↑
 Char:    ←  →     Char:   Ins    Char:   Del    Down a field:  ↓
 Word: Home End    Field:  ^N     Word:   ^Y     Exit/Save:  ^End
 Pan:     ^←  ^→   Help:   F1     Field:  ^U     Abort:       Esc

   Field Name   Type        Width  Dec

1  CHECKNO      Numeric        4    0
2  REASON       Character     48
3  TO_WHOM      Character     48
4  AMOUNT       Numeric       12    2
5  DATE         Date           8
6               Character

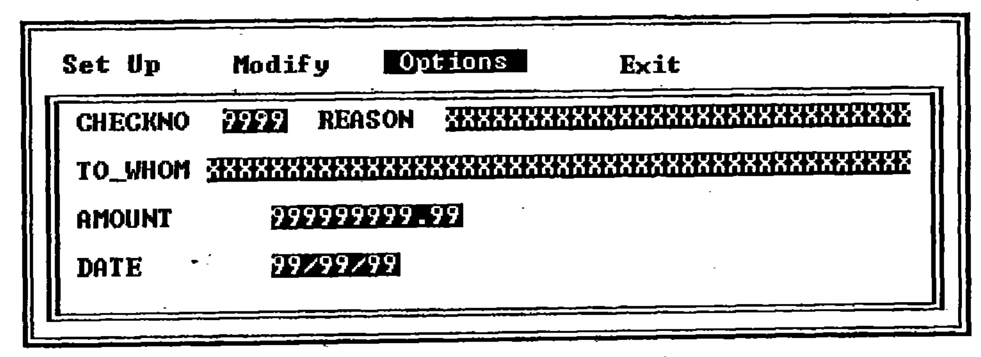
 CREATE      |<C:>|CHECKS              |Field: 6/6  |    |Num
                      Enter the field name.
```

شكل (٢٨ - ١)

ويمكن عمل نفس الشيء مع الإختيارات (2) , (3) , (4). فاذا أراد المستخدم استخدام شاشة ادخال بدلا من شاشة الادخال المبدئية الخاصة ببرنامج (+ DBase III) فإنه يختار الرقم (2). وفى هذه الحالة يظهر السؤال التالى على الشاشة.

Enter screen file name :

ويتم إدخال إسم الشاشة وليكن أيضا (CHECKS) ثم الضغط على مفتاح الادخال وفى هذا الحالة تظهر شاشة الرسم التى يمكن من خلالها اختيار (Select a database) وكتابة اسم قاعدة البيانات (CHECKS). ثم يختار المستخدم (Load fields). والشكل (٢٨ - ٢) يوضح شاشة الادخال بعد تصميمها. ثم يختار المستخدم (Save) من قائمة (Exit) لتعود قائمة مولد التطبيقات إلى الظهور.



شكل (٢٨ - ٢)

والاختيار الثالث فى قائمة مولد التطبيقات يساعد المستخدم على تصميم التقرير من خلال مولد التقارير الخاص ببرنامج (+ DBase III). فعندما يختار المستخدم الرقم(3) من القائمة يظهر سؤال عن اسم نموذج التقرير المطلوب فيتم كتابة (CHECKS). ويقوم المستخدم بكتابة توصيف التقرير كالآتى مثلا :

```
Contents              REASON + TO - WHOM
Heading               Reason and ; To whom Written
Width                 30
Decimall Places
Total this Column
Report Format
>>>>>>>>>             Reason and         Amount      Date
                      To Whom Written
                      xxxxxxxxxxxxxxxx ########     mm/dd/yy
```

ويقوم المستخدم بعد ذلك بتخزين التقرير ثم تعود قائمة مولد التطبيقات إلى الظهور.

والاختيار الرابع فى القائمة يتيح للمستخدم تصميم نموذج العناوين البريدية (Mailing Labels) ولكن ليست هناك حاجة إلى ذلك فى هذا التطبيق. والإختيار الخامس يتيح للمستخدم تحديد ألوان الشاشات الخاصة بالتطبيق. ويؤدى هذا الاختيار إلى ظهور الشاشة الموضحة بالشكل (٢٨ - ٣)

ويتم من خلال هذه الشاشة اختيار الألوان المطلوبة ثم يتم الضغط على مفتاحى (End ^) ثم كتابة (S) لتخزين الألوان.

٢٨ - ٢ التوليد الآلى للبرنامج

بعد إنشاء قاعدة البيانات وشاشة الادخال ونموذج التقرير يتم اختيار الرقم (6) من القائمة لتوليد التطبيق المطلوب. وفى هذه الحالة يظهر الآتى على الشاشة

Enter Application Name :

فيتم إدخال إسم التطبيق المطلوب وليكن (Register) ثم يظهر الآتى على الشاشة

Enter Database Filename :

```
set color                                                    01/13/86

              1. Black              6. Red
              2. Blue               7. Magenta
              3. Green              8. Brown
              4. Cyam               9. White
              5. Blank (Secure)

                                                       BRIGHT  BLINNING
Enter COLOR number for LETTERS in STANDARD TEXT :  G           N
Enter COLOR number for STANDARD BACKGROUND :       I           N
Enter COLOR number for LETTERS in ENHANCED TEXT :  I           N
Enter COLOR number for ENHANCED BACKGROUND :       G           N
Enter COLOR number for OIRDER                      B           N
Command | < C:> | CHECKS | Rec: None | Ins |
```

شكل (٢٨ - ٣)

فيقوم المستخدم بكتابة إسم ملف قاعدة البيانات (CHECKS). ثم يسأل البرنامج عن إسم ملف الفهرس فيتم كتابة (CHECKS). ثم يسأل البرنامج عن حقل المفتاح المستخدم فى الفهرس فيتم كتابة (CHECKNO) إذا كان مطلوبا ترتيب قاعدة البيانات بناء على رقم الشيك. ثم يسأل البرنامج عن إسم شاشة الادخال فيتم كتابة إسم ملف شاشة الإدخال الذى سبق إنشاؤه وهو (CHECKS). ثم يسأل البرنامج عن إسم نموذج العناوين البريدية (Mailing Labels) وحيث أنه لم يتم تصميم هذا النموذج فيتم الضغط على مفتاح الإدخال دون كتابة أى إسم. ثم يسأل البرنامج عن عنوان التطبيق فيتم كتابة (CHECKBOOK REGISTER) والضغط على مفتاح الإدخال. وبعد الإنتظار قليلا يقوم البرنامج بإنشاء التطبيق المطلوب كما يقوم بعرض الكود الذى تم إنشاؤه. وذلك كما يتضح من السطور التالية :

```
* Program - - - : CHECK.PRG
*  Author - - -  :  Ahmed Maher
*  Date - - - - - :  01/13/1992
*  Notes - - - -  :
*  Reserved - - :  Selectnum
```

```
SET TALK OFF
SET BELL OFF
SET STATUS  ON
SET ESCAPE  OFF
SET CONFIRM ON
USE CHECKS INDEX CHECKS
DO WHILE  .T.
        * - - - Display menu options centered on the screen.
        * - - - Draw menu border and print heading.
        CLEAR
        @  2,0  TO 15.79  DOUBLE
        @  3,22  SAY ( CHECK BOOK REGISTER )
        * - - - Display detail lines.
        @  7,30   SAY ( 1.  ADD INFORMATION )
        @  8,30   SAY ( 2.  CHANGE INFORMATION )
        @  9,30   SAY ( 3.  REMOVE INFORMATION )
        @  10,30  SAY (4 .  REVIEW INFORMATION )
        @  11,30  SAY ( 5.  PRINT INFORMATION )
        @  13,30  SAY '0 . EXIT '
        STORE  0 TO Selectnum
        @  15,33  SAY " select "
        @  15,42  GET selectnum PICTURE "9" RENGE 0,5
        READ
        DO CASE
                CASE Selectnum = 0
                        SET BELL ON
                        SET TALK ON
                        CLEAR ALL
                        RETURN
                CASE selectnum = 1
                        * DO ADD INFORMATION
                        SET FORMAT TO CHECKS
```

```
        APPEND
        SET FORMAT TO
        SET CONFIRM OFF
        STORE '   ' TO wait _ subst
        @ 23,0 SAY'Press any key to continue.' ;
           GET wait_subst
        READ
        SET CONFIRM ON
CASE selectnum = 2
        *  DO CHANGE INFORMATION
        SET FORMAT TO CHECKS
        EDIT
        SET FORMATTO
        SET CONFIRM OFF
        STORE '  ' TO  wait _ subsf
        @ 23,0 SAY 'Press any key to continue'  ;
        GET wait_subst
        READ
        SET FORMAT ON
CASE selectnum = 3
        *  DO REMOVE  INFORMATION
        SET TALK ON
        CLEAR
        @ 2,0 SAY '  '
        ? ' PACKING DATABASE '
        PACK
        SET TALK OFF
        SET CONFIRM OFF
        STORE '  ' TO  wait_ subst
        @ 23,0 SAY ' Press any key to continue' ;
        GET  wait_subst
        READ
```

```
            SET FORMAT ON

        CASE selectnum = 4
            * DO REVIEW  INFORMATION
            BROWSE
            SET CONFIRM OFF
            STORE ' ' TO  wait _ subst
            @ 23,0 SAY 'Press any key to continue ..' ;
            GET wait_subst
            ERAD
            SET CONFIRM ON
    ENDCASE
ENDDO T
REUTURN
* EOF : CHECK.PRG
```

٢٨ - ٣ تشغيل البرنامج التطبيقى

لتشغيل البرنامج الذى تم تصميمه يختار المستخدم الإختيار (7) من قائمة مولد التطبيقات وفى هذه الحالة تظهر القائمة التالية على الشاشة :

1. ADD INFORMATION
2. CHANGE INFORMATION
3. REMOVE INFORMATION
4. REVIEW INFORMATION
5. PRINT REPORT
0. EXIT

والاختيار (1) يتيح للمستخدم إضافة سجلات جديدة إلى قاعدة البيانات. والاختيار (2) يتيح له إجراء تعديلات على قاعدة البيانات ويمكن إستخدام مفتاحى (PgUp) ، (PgDn) للوصول إلى السجل المطلوب تعديله. والإختيار (4) يتيح للمستخدم إستعمال العرض الجدولى (BROWSE) فى تعديل البيانات ومسحها بإستخدام مفتاحى (U ^).

والإختيار (3) يؤدى إلى مسح السجلات نهائيا باستخدام الأمر (PACK). والإختيار (5) يؤدى إلى طباعة التقرير. والإختيار (0) يؤدى إلى الخروج من النظام والعودة إلى قائمة مولد التطبيقات.

ويمكن تشغيل البرنامج التطبيقى مباشرة من خلال مشيرة النقطة (Dot Prompt). ولكن يلزم فى الحالة أن تكون جميع الملفات التى تم إنشاؤها على نفس القرص أو نفس الفهرس الفرعى بالإضافة إلى الملفين الذين قام مولد التطبيقات بانشائهما وهما (REGISTER.PRG) ، (REGISTER.MEM). ويتم تشغيل البرنامج فى هذه الحالة بكتابة السطر التالى :

DO REGISTER

ثم الضغط على مفتاح الإدخال.

٢٨ - ٤ مولد التطبيقات المتقدم

يمكن استخدام مولد التطبيقات المتقدم عن طريق الإختيار رقم (8). وفى هذه الحالة يسأل البرنامج نفس الأسئلة السابقة ولكنه بالإضافة إلى ذلك يعرض شاشة تتيح للمستخدم إدخال إختيارات القوائم التى يريد استخدامها خلال البرنامج مع إدخال الأوامر التى يريد استخدامها عند كل اختيار وفى النهاية يقوم بتخزين البرنامج ثم تشغيله بالطريقة السابقة.

٢٨ - ٥ تعديل البرنامج التطبيقى

يستطيع المستخدم فى أى وقت تعديل كود البرنامج التطبيقى الذى تم انشاؤه بواسطة مولد التطبيقات. ولتنفيذ ذلك فإنه يختار الإختيار رقم (٩) من قائمة مولد التطبيقات وهذا يؤدى إلى تحميل كود البرنامج فى المصحح (Editor) الخاص ببرنامج (DBase III +) ثم يستطيع تعديل سطور البرنامج كما يريد.

الملاحق

ملحق (١)

مجموعة كتب دلتا

١ الحاسبات الإلكترونية حاضرها ومستقبلها

يعتبر هذا الكتاب من أهم الكتب التى يحتاج القارىء اليها سواء كان فى بداية طريق دراسة علوم الحاسب أو قطع شوطا كبيرا فى هذا المجال . ذلك لأن هذا الكتاب يتضمن معلومات عن كل مايتعلق بتكنولوجيا الحاسب بدءا من إستعراض تطور الحاسبات من حيث المكونات المادية والبرامج وتطور نظم التشغيل وانتهاء بلغات الجيل الرابع ونظم دعم القرار مرورا بجميع الموضوعات التى تشغل المتخصصين فى مجال الحاسب مثل تعريب الحاسبات ولغات الحاسب بالإضافة إلى البرامج التطبيقية المختلفة مثل نظم إدارة قواعد البيانات والجداول الإلكترونية وبرامج تنسيق الكلمات ونظم إدارة المشروعات ونظم التصميم الهندسى هذا بالإضافة إلى موضوعات أمن البيانات وفيروسات الحاسب . ويحتوى هذا الكتاب على جزء خاص بمستقبل تكنولوجيا الحاسبات يتضمن الذكاء الإصطناعى والنظم الخبيرة والبرمجة الشيئية (Object Oriented) والمعالجة العصبية للمعلومات (Neural Networks) ومعالجة اللغات الطبيعية واللغة العربية بالحاسب والكثير من الموضوعات الأخرى المرتبطة بهذا المجال . والكتاب يحتوى على مالايقل عن ٨٥٠ صفحة مزودة بأكثر من ٥٠٠ شكل توضيحى .

٢ دائرة معارف الحاسب الإلكترونى

يعتبر هذا الكتاب من المراجع العلمية المتميزة فى مجال تكنولوجيا المعلومات. فهو إلى جانب مايتمتع به من دقة وشمول فإن أسلوبه يتميز بالسهولة والوضوح دون الإخلال بالمضمون العلمى. والكتاب لايقتصر على الترجمة الدقيقة لمصطلحات الحاسب ، وإنما يوفر أيضا الشرح التفصيلى لهذه المصطلحات وأى معلومات مرتبطة بها. وقد روعى عند إعداد هذه الموسوعة أن يجد فيها القارئ كل غايته ، بدءا من القارئ العادى الذى يسعى إلى الحصول على المعلومات البسيطة الشاملة ، وانتهاء بالقارئ الفنى والمتخصص الذى يسعى إلى الحصول على

معلومات فنية دقيقة. لذلك فقد تم تغذية الموسوعة بآخر ماوصل إليه العلم فى مجال تكنولوجيا المعلومات لملاحقة التطور السريع فى هذا المجال. ويحتوى الكتاب على مالايقل عن ألف ومائتى مصطلح مرتبة بالترتيب الهجائى للحروف حتى يستطيع القارئ بسهولة الوصول إلى المصطلح المطلوب. ويصل عدد صفحات الكتاب إلى ٥٠٠ صفحة مزودة بما يزيد عن ٣٠٠ شكل توضيحى

٣ المرجع الشامل لنظام التشــــغيل (DOS)

يعتبر هذا الكتاب من المراجع العربية المتميزة التى تتناول نظام التشغيل (DOS) وتوضح خصائصه الفنية وأوامره ووظائفه بشرح يتصف بالبساطة إلى جانب الدقة والشمول . والكتاب لايقتصر على نظام التشغيل (DOS) فقط ، ولكنه يتناول أيضا نظام التشغيل (DOS-4)، (DOS-5) ، (DRDOS-6) بالإضافة إلى نظام النوافذ (Windows) الذى يوفر التفاعل الجيد بين المستخدم والحاسب . كما يتناول الكتاب أيضا أهم الأدوات المساعدة لنظام التشــــغيل (DOS) مثل برنامــــج (PC Tools) وبرنامج(Norton) . وهى الأدوات التى تساعد المستخدم على إستعادة الملفات الممسوحة بطريق الخطأ وكذلك فحص القرص واكتشاف أعطاله وإصلاحها وتحسين أداء القرص وتحسين أداء الحاسب بصفة عامة . كما يتناول الكتاب أيضا أهم السلبيات والمشاكل التى يمكن أن يتعرض لها نظام التشغيل (DOS) ممثلة فى فيروسات الحاسب مع توضيح أخطار هذه الفيروسات وطرق التغلب عليها والوقاية منها . والكتاب يتكون من ستة أجزاء بالإضافة إلى الملاحق ، و يزيد عدد صفحاته عن ٦٥٠ صفحة محتوية على مايزيد عن ٦٠٠ شــكل توضيحى .

٤ عالـــم الجداول الإلكترونيــة (بين الدراسة والتطبيق)

يعتبر هذا الكتاب من أهم الكتب التى تناولت برامج الجداول الإلكترونية بالشرح التفصيلى الدقـيق مع الأسلوب السـهل الواضح . ورغم أنه يشــرح ثلاثة من البــرامج تمثل أقــوى

برامج الجداول الإلكترونية على الإطلاق وهى برامج :

LOTUS 123 - EXCEL - QUATRO PRO

إلا أنه يتضمن أيضا شرحا وافيا لأساسيات التعامل مع برامج الجداول الإلكترونية بصفة عامه مما يساعد المستخدم على الإلمام بأساليب التعامل مع جميع برامج الجداول الإلكترونية . ويوفر البرنامج شرحا لأهم الخصائص الفنية المتقدمة مثل استخدام الماكرو واستخدام خصائص قواعد البيانات واستخدام النوافذ وربط الجداول الإلكترونية واستخدام مكتبات الربط وكذلك استخدام الأنواع المتقدمة من الرسومات مثل الرسومات ثلاثية الأبعاد واستخدام الشاشات المنزلقة . كما يوفر الكتاب شرحا تفصيليا لطريقة حل مسائل البرمجة الخطية عن طريق الجدول الإلكترونى مع توضيح ذلك بمثال عملى واضح . كما يشرح الكتاب تطبيقا شاملا على الجداول الالكترونية وهو بعنوان "إدارة التدفق النقدى" وذلك فى اكثر من ٢٥٠ صفحة متضمنة عددا كبيرا من الرسوم التوضيحية والمخططات . وهذا التطبيق يساعد مدير العمل على متابعة التدفق النقدى فى منشأته والسيطرة عليه . والكتاب فى مجمله يزيد عدد صفحاته عن ٦٧٠ صفحة متضمنة مالا يقل عن ٦٠٠ شكل توضيحى .

٥ الحاسب الإلكترونى وقواعد البيانات (الجزء الأول)

يعتبر هذا الكتاب من أهم الكتب التى تتناول قواعد البيانات بصفة عامة وبرامج عائلة (DBase) بصفة خاصة وهى البرامج :

DBASEIII+ - DBASEIV - FOXBASE+ - FOXPRO

والكتاب يوضح مفهوم قواعد البيانات ومفهوم إدارة قواعد البيانات . كما يشرح الكتاب بالتفصيل أهم الجوانب الفنية المرتبطة ببرامج عائلة (DBase) متضمنة قواعـــد إنشــاء هيكل الملف (DBase Structure) وقواعد تصميم شاشة الإدخال وعرض السجلات على الشاشة وتصحيحها وقواعد تنظيم ملف قاعدة البيانات عن طريق الفرز (Sorting) والفهرسة

(Indexing) وطـــرق البحث عـــن الســجلات واستخدام ملفات البحث (Query Files) وطباعة التقــارير وربط قـواعد البـيانات، كما يشـرح الكـتـاب اسـتخدام أوامر النقطة (Dot Commands) وقواعد كـتابـة البرامـج الخـاصـة بقواعد البيانات وإستخدام متــغيرات الذاكـــرة (Memory Variables) وملفات الذاكرة (Memory Files) وملفات الخطوات (Procedure Files) والدوال المستخدمة وطرق التحكم في شاشة الإدخـــال وكذلك التحكم فـــي الطباعة ووسائل تصحيح الأخطاء (Debugging Tools) . ويتكـــون الكتـــاب مـن ستة وعشرين فصلا بالإضافة إلى أربعة ملاحق.كما يحتوى على العديد من الأشكال التوضيـــحية ويزيد عدد صفحاته عن ٣٨٠ صفحة.

٦ الحاسب الإلكتروني وقواعـــد البيانات (الجزء الثاني)

يعتبرهذا الكتاب جزءا مكملا للجزء الأول ويحتوى على شرح تفصيلي للأوامر والدوال المستخدمة في برامج عائلة (DBase) و يكون مع الجـــزء الأول المرجع الشــامل الذي يعين المستخدم على كتابة البرامج التطبيقية عالية الكفاءة والتي تخدم جميع مجالات نظم المعلومات . ويزيـــد عـــدد صفحـــات الكتـــاب عن ٢٤٠ صفحـــة متضمنـــة العديـــد مـــن الأشكال التوضيحية .

٧ تطبيقات نظم إدارة قواعـــد البيانـــات

يعد هذا الكتاب إضافة حقيقية للمكتبة العربية التي تفتقر إلى هذا النوع من الكتب التي تتناول تطبيقات عملية لنظم إدارة قواعد البيانات . ولا يكتفي الكتاب بالشرح الإجمالي لكل نظام والبرامج المكونة له ، ولكنه يقف عند كل سطر في البرامج ويشرحه شرحا دقيقا موضحا البدائل المختلفة ومميزات وعيوب كل من هذه البدائل . والكتاب يتكون من ستة أجزاء . الجـــزء الأول يحتوى على مراجعة للكتاب الأول "نظم إدارة قواعد البيانات" بجزأيه الأول والثاني . والجزء الثاني من الكتاب يشرح نظام معلومات شئون الطلبة الذي يصلح للاستخدام في أي

مؤسسة تعليمية لمتابعة بيانات الطلبة والسيطرة الكاملة على إدخال البيانات وعرضها وتصحيحها وطباعة التقارير . والجزء الثالث من الكتاب يشرح نظام المخازن كنموذج لقواعد البيانات التى تتعامل مع ملفات الحركة (Transaction Files) . والجزء الرابع يشرح نظام حسابات العملاء كنموذج للبرامج التى تستخدم ملفات الخطوات (Procedure Files) لتقليل عدد الملفات المفتوحة . والجزء الخامس يشرح بعض الأدوات والوسائل المتقدمة فى كتابة البرامج من خلال ثلاثة برامج مختلفة أحدها يستخدم فى كتابة الشيكات ، والثانى يتيح للمستخدم إختيار الألوان التى يفضلها فى شاشات إدخال البيانات ، والثالث يتيح عرض شاشات إدخال بيانات تحتوى على عمود ضوئى يمكن تحريكه الى الإختيار المطلوب والجزء السادس يشرح بعض التطبيقات الإضافية ويتضمن أيضا شرح مولد التطبيقات الخاص ببرنامج (DBase III+). والكتاب يزيد عدد صفحاته عن خمسمائة صفحة متضمنة مايزيد عن ١٠٠ شكلا توضيحيا .

٨ فيروسات الحاسب وأمن البيانات

يتناول هذا الكتاب قضية أمن البيانات بصفة عامة موضحا الأساليب التكنولوجية المختلفة لتنفيذ ذلك مثل استخدام التشفير وإعادة التشفير واستخدام كلمات السر تبعا لمستويات السرية المختلفة وارتباط ذلك بنظم التشغيل. ثم يشرح الكتاب موضوع فيروسات الحاسب باعتباره من أهم الموضوعات التى تشغل عقول كثير من المهتمين بمجال الحاسب نظرا لما يمثله الفيروس من خطورة على أمن البيانات . ويقدم الكتاب دراسة موضوعية دقيقة تتناول التحليل الدقيق للفيروس من حيث تكوينه وخصائصه الفنية بما يتيح للمستخدم التعرف السليم على هذا الموضوع بعيدا عن التخيلات والأوهام . كما يشمل الكتاب أيضا توضيحا لطرق الوقاية والعلاج والأمصال البرمجية المستخدمة ضد أنواع معينة من الفيروسات . ويتضمن الكتاب بعض الملاحق يناقش أحدها أشهر نماذج فيروسات الحاسب مع شرح دقيق لمواصفات أكثر من ١٥٠ فيروس من فيروسات الحاسب .

٩ الحاسب ونظم المعلومات الإدارية

يتناول هذا الكتاب أهمية إستخدام الحاسب فى نظم المعلومات الإدارية وأهم الموضوعات المرتبطة بتكنولوجيا المعلومات وتطور الحاسبات ونظم التشغيل ولغات الحاسب . كما يوضح أساسيات تحليل وتصميم النظم بدءا من توصيف المتطلبات وتحليلها ثم تحليل بدائل تصميم النظام وكذلك استخدام أدوات هندسة البرامج (CASE Tools) . كما يتناول أهم تطبيقات الحاسب المالية والمحاسبية مثل نظام السيطرة على المخزون ونظم حسابات العملاء والحسابات العامة والمرتبات وإدارة التدفق النقدى والتسويق والتصنيع . كما يتناول الكتاب استخدام التقنيات الحديثة فى مجالات الأعمال مثل ميكنة المكاتب والإتصالات وأمن البيانات ونظم المعاونة فى اتخاذ القرار والذكاء الإصطناعى . والكتاب يزيد عدد صفحاته عن ٥٥٠ صفحة تتضمن مايزيد عن ٤٠٠ شكل توضيحى .

١٠ النظم الخبيرة والذكاء الإصطناعى

يتناول هذا الكتاب تقنية من أحدث التقنيات التى ظهرت فى عصر الحاسب ، وهى التقنية الخاصة بالذكاء الإصطناعى مع تناول أحد المجالات التطبيقية الهامة المرتبطة بها بالتفصيل وهى النظم الخبيرة والتى بدأت تنتشر بسرعة كبيرة فى معظم أوجه الحياة العملية . وقد وضع فى الإعتبار أن يجد كل من القارئ المتخصص وغير المتخصص غايته من هذا الكتاب بحيث يتمكن القارئ من التفاعل بسلاسة وسرعة مع تكنولوجيا المستقبل . والكتاب يزيد عدد صفحاته عن ٣٥٠ صفحة متضمنة العديد من الأشكال التوضيحية .

رقم الإيداع : ٩١/٢٨٨٥

مطابع المكتب المصرى الحديث
MODERN EGYPTIAN PRESS
ت : ٢٢١١٠٧١ - ٢٢١١٠٧٢ - فاكس ٢٢١١٠٧٣

مؤسسة " دلتا "

تعتبر مؤسسة " دلتا " من المكاتب الاستشارية الرائدة ذات الخبرات الفنية والعلمية الرفيعة والامكانات المتكاملة والمتميزة بتعدد التخصصات والخبرات فى نظم المعلومات الآلية .

وتتكون المؤسسة من عدد كبير من المتخصصين ذوى الخبرات الواسعة والعلميين من أساتذة الجامعات الممارسين للعديد من الحقول الفنية والبحثية المرتبطة بمجالات نظم المعلومات والاتجاهات المتطورة لميكنتها . فقد إتخذت المؤسسة الأساليب العلمية منهاجا فى تقديم الحلول للمشاكل المتنوعة والمعقدة والتى طالما تواجه العديد من المشروعات .

وتعتنى الدراسات الفنية التى يقوم بها خبراء المؤسسة بالعمل على تطوير الوسائل المناسبة للاستفادة من التكنولوجيا الحديثة فى مجالات نظم المعلومات . ويجدر الاشارة هنا بأن خبراءنا يشغلون العديد من المناصب القيادية ويقدمون الاستشارات العلمية والفنية للعديد من الهيئات والمؤسسات وحيث تجاوز مجال أنشطتهم الحدود المصرية الى المنطقة العربية كما يشغل بعض أعضاء المؤسسة مراكز أساسية فى اللجان الفنية الوطنية والعالمية فى الأعمال التى تتعلق بتخصصاتهم .

وعلى مدى أكثر من عشر سنوات قام خبراء ومستشارو مؤسسة " دلتا " بتحقيق العديد من الانجازات التى يمكن عرض بعض اتجاهاتها فيما يلى :

١ - القيام بدراسات الجدوى لادخال نظم الحاسبات الآلية فى الهيئات والمؤسسات المختلفة .

٢ - تحليل وتصميم وتنفيذ العديد من النظم الآلية والاشراف على المشروعات .

٣ - تصميم وتنفيذ البرامج التطبيقية للحاسبات الآلية فى العديد من مجالات نظم المعلومات والشئون المالية والادارية .

٤ - عمل الدراسات الخاصة بتقييم مستويات الأداء للنظم الآلية مع تحديد أساليب تطويرها .

٥ - تنفيذ برامج التدريب المتطورة على النظم الآلية المتخصصة .

٦ - القيام بالعديد من الأبحاث العلمية التى تتناول تعريب الحاسبات والقيام بالانجازات التطبيقية فى هذا المجال .

وأخيرا وليس آخرا فان مؤسسة " دلتا " قد أخذت على عاتقها مهمة اصدار سلسلة المراجع المتخصصة فى مجال تكنولوجيا وعلوم الحاسب حتى يستفيد منها أكبر عدد من القراء المتخصصين بالاضافة الى العديد من الدارسين فى مصر والعالم العربى .

والله الموفق ،،،،

مجموعة كتب "دلتا"



هذا الكتاب

١ - مقدمة

الجزء الأول

نظم إدارة قواعد البيانات

٢ - انواع البرامج
٣ - البحث السريع
٤ - خطوات تصميم النظام
٥ - كتابة البرامج
٦ - وسائل التصحيح

الجزء الثانى

نظام معلومات الطلبة

٧ - تصميم النظام
٨ - البرنامج الرئيسى
٩ - برنامج التقارير
١٠ - برنامج التصحيح
١١ - برنامج مسح السجلات

الجزء الثالث

نظام الخان

١٢ - توصيف النظام
١٣ - برنامج القائمة الرئيسية
١٤ - برنامج تشغيل الملف الرئيسى
١٥ - برنامج تشغيل ملف المبيعات
١٦ - برنامج تشغيل ملف الاضافة
١٧ - برنامج تحديث البيانات

الجزء الرابع

نظام حسابات العملاء

١٨ - تصميم النظام
١٩ - ملفات الخطوات
٢٠ - برنامج القائمة الرئيسية والادخال والتعديل
٢١ - تقارير برنامج حسابات العملاء
٢٢ - التحديث الشهرى للنظام
٢٣ - برنامج التكامل بين حسابات العملاء والمخازن

الجزء الخامس

بعض الأدوات المتقدمة

٢٤ - برنامج كتابة الشيكات
٢٥ - برنامج اختيار الألوان
٢٦ - برنامج تحريك العمود الضوئى

الجزء السادس

تطبيقات إضافية

٢٧ - التطبيق المنزلى
٢٨ - مولد التطبيقات

الملاحق

ملحق (١) مجموعة كتب دلتا

٥ عبد الله بن الزبير مصر الجديدة - نزهة - القاهرة - ت . ٢٤٦٧٣٣٨ / ٢٤٤٠٣٧٥ / ٢٤٤٤٩٩٧ فاكسيملى ٢٤٣٥٦٦٥
5 ABDULLAH EBN EL-ZUBAIR NOZHA HELIOPOLIS, CAIRO Fax. 2435665 Tel.: 2467338/2440375/2444997